

REMARKS

The application has been amended and is believed to be in condition for allowance.

The Official Action indicated that the drawings filed in the application were acceptable for examination but would need to be replaced with formal drawings when the application is allowed. Formal drawings are attached.

The Official Action stated that a substitute specification including the claims was required pursuant to 37 CFR 1.125(a). No more specific basis for requiring a substitute specification is otherwise given.

37 CFR 1.125(a) requires a substitute specification if the "number or nature of amendment or the legibility of the application papers renders it difficult to consider the application, or to arrange the papers for printing or copying,..."

There has only been a few minor amendments to the specification. Therefore, applicants understand that their basis for the substitute specification requirement is that the filed papers are deemed of insufficient legibility. Attached is a substitute specification including the amendments of record as well as formal amendments to improve readability. The undersigned attorney verifies that there is no new matter being entered in the substitute specification.

The Official Action indicates that the specification must be in proper idiomatic English and in compliance with 37 CFR 1.52(a) and (b). The substitute specification is believed to meet these requirements.

As to priority, note that this application is a National Stage of a PCT application filed pursuant to 35 USC 371. The PCT properly claims priority to applicants' European application filed of October 13, 1997 as the PCT was timely filed on October 13, 1998.

Claims 28, 37, and 56 are rejected as indefinite for including the phrase "at least."

Claims 30 and 39 are rejected as indefinite for the phrase "like...".

Claim 46 is rejected as indefinite for the phrase "e.g.".

Each of the noted claims have been amended to remedy the stated basis of rejection. Claim 28 has been cancelled and replaced with claim 60 drafted to be proper as to form and more clearly recite the invention. Withdrawal of these rejections is solicited.

New claims 57-59 include recitations cancelled from claims 30, 39, and 46 respectively.

New claim 61, depending from claim 60, recites that the sender identifier and the destination address are each email

addresses in the form of a user name and a domain name, e.g., user-zyz@ilserver.com. Original specification page 17, line 25 supports this recitation.

Claims 28-56 were objected to as "the referenced claims are replete with spelling and grammar mistakes."

The undersigned attorney has reviewed these claims, aided by a word processing spelling checker, to find spelling errors. Should there be spelling errors that escaped detection, please advise and the errors will be promptly corrected.

As to grammar, the undersigned attorney has also reviewed the claims and has made formal amendments that are designed to improve the grammar. Should any further amendments be necessary, please advise with specificity.

Withdrawal of all the formal objections/rejections is solicited.

Claims 28-37 and 56 stand rejected as obvious over SAMUEL et al. 6,018,766 ("SAMUEL").

Claims 38-55 stand rejected as obvious over SAMUEL et al. in view of COCHINWALA et al. 5,960,178.

Consider prior claim 28 first.

Claim 28 recited a method of **point-to-point communication** (between a sender and a receiver by messages with flexible message formats) through the step of **conducting point-to-point communications** by interpreting and processing messages using

a database (ILMDB) storing a message definition table (msgdef), a field definition table (flddef), mapping instructions (fldmap) and message action lists (fldact, msgpre, msgpost).

As an initial matter, SAMUEL does not teach a point-to-point communication method but rather teaches a group messaging technique for sending messages from one sender to a group of receivers, using unicast network architecture. See the Abstract disclosing a method where the hosts send messages containing destination group addresses by unicast to the group messaging servers, and each group messaging server in turn forwards the messages to each of the target hosts within its group.

Claim 28 further recited that the messages each comprises a header and a message content portion. Claim 28 requires that the header comprise message definition references (MSG ID, MSG CLASS, MSG VERSION, MSG CREATOR), a sender identifier (SENDER ID) and a destination address (DESTINATION ADDRESS), and that the message content include:

i) a number of fields (FIELD COUNT) and a content of any field (FIELD(1),...),

ii) a number of objects (OBJECT COUNT) and a content of any object (OBJECT(1),...), the objects being referred to by one or more of the fields,

iii) a number of field mappings and a content of any field mapping, any field mapping being usable by predetermined fields, and

iv) a number of actions and a content of any actions, any action being usable by the predetermined fields.

In rejecting claim 28, the Official Action relies on SAMUEL Figure 9 and the text explaining the figure. Figure 9 is a detailed datagram format and address format for ULP (upper level protocol) messages of SAMUEL. Reference is made to the specification beginning at line 58 of column 13.

The SAMUEL ULP datagram protocol encapsulates addresses, message type information and the message payload within a datagram of the underlying network transport protocol.

Figure 9, the first line of elements illustrate the SAMUEL ULP datagram message format, which format comprises elements 123, 124, 125, 126, 127, 128 and 129, where:

transport header 123 is the datagram header of the TLP (transport level protocol) that is encapsulating the ULP datagram;

ULP message type field 124 indicates message type, e.g., send or receive message, control message or state message (See that SAMUEL defines send messages as those sent from a host to a group messaging server, whereas messages from a group server to the hosts are receive messages. Send control messages are

messages from hosts to a group messaging server requesting a control function be performed);

destination ULP address 125 specifies the primary destination of the ULP message;

address count field 126 (used in send messages but not in receive messages), when non-zero, specifies the number of auxiliary destination addresses for the send message that follow the address count field;

127-128 are the auxiliary destination addresses; and  
129 is the payload.

Figure 9, the second line of elements illustrates the payload (element 129 of the ULP message format). The payload format is defined by items 116, 117, 118, 119, 120, 121 and 122 (column 14, beginning at line 37). Item 116 is the message count defining how many payload elements will be contained in the payload. Each single payload element consists of a triplet of: source ULP address, data length and data, e.g., items 117, 118 and 119 comprise the first payload element of the payload. Item 117 is the ULP address of the source of the payload element, item 118 is the data length for the data in the payload element and item 119 is the actual data. Items 120, 121 and 122 comprise the last payload element in the payload.

The address space of the ULP is divided into three segments: unicast host addresses, implicit group addresses and

logical group addresses. The format for ULP addresses is shown in the last line of Figure 9 comprised of items 130, 131 and 132.

Paragraph 18 of the Official Action states that SAMUEL teaches a point-to-point communication between a sender and a receiver. As noted above, SAMUEL does not teach point-to-point communication.

Claim 28 recited a header and a message content portion.

For the recitations of the header comprising message definition references (MSG ID, MSG CLASS, MSG VERSION, MSG CREATOR), a sender identifier (SENDER ID) and a destination address (DESTINATION ADDRESS), SAMUEL Figure 9 elements 117, 123, and 125 were offered. The Official Action failed to indicate which recited features were believed to read on which of these three elements.

Element 125 is the destination ULP address 125 specifying the primary destination of the ULP message, and would seem to being read on the recited destination address.

That would leave 117 and 123 to read on the message definition references and sender identifier.

Element 117 is shown in the second line of Figure 9. As discussed above, the second line of elements illustrates the payload (element 129 of the ULP message format shown in the first line). The payload format is defined by item 116, and triplets

of 117, 118, and 119 (column 14, beginning at line 37). As discussed above, item 117 is the source ULP address of the triplet (item 118 is the data length for the data in the payload element and item 119 is the actual data). Thus, 117 would seem to be offered for the "sender identifier (SENDER ID)".

Note, however, that 117 is not part of the message header but is part of the message payload. Thus, 117 cannot anticipate the recitation of a sender identifier within the header.

Also 117 cannot be the recited message definition references (MSG ID, MSG CLASS, MSG VERSION, MSG CREATOR). Again, 117 is not part of the header and does not include message definition references.

Lastly, 117 is clearly not the destination address.

Element 123 is "transport header 123", disclosed as the datagram header of the TLP (transport level protocol) that is encapsulating the ULP datagram. The nature of element 123 is not detailed; however, there is no disclosure that this element meets any of the recited message definition references (MSG ID, MSG CLASS, MSG VERSION, MSG CREATOR), sender identifier (SENDER ID) or destination address (DESTINATION ADDRESS).

Thus, at least two of the header components have not been identified as being disclosed by SAMUEL.



As noted above, the message content portion includes i) a number of fields (FIELD COUNT) and a content of any field (FIELD(1),...); ii) a number of objects (OBJECT COUNT) and a content of any object (OBJECT(1),...), the objects being referred to by one or more of the fields; iii) a number of field mappings and a content of any field mapping, any field mapping being usable by predetermined fields; and iv) a number of actions and a content of any actions, any action being usable by the predetermined fields.

For the message content portion i) a number of fields (FIELD COUNT) and a content of any field (FIELD(1),...), SAMUEL Figure 9, element 116 was offered. Although part of the payload 129 element, element 116 only defines how many payload elements (comprised of triplets 117-119) will be contained in the payload, this is at best only part of the recitation. Even if the number of payload elements is read on number of fields, element 116 does not include the content of each field. That is, element 116 does not include the content of each payload element.

Next blocks 117, 118, 119 have been offered for the recitation of the message content portion including ii) a number of objects (OBJECT COUNT) and a content of any object (OBJECT(1),...), the objects being referred to by one or more of the fields. SAMUEL payload 129 comprises one of more payload element triplets of 117, 118, and 119 (column 14, beginning at

line 37). As discussed above, item 117 is the source ULP address of the triplet, item 118 is the data length for the data in the particular payload element, and item 119 is the actual data.

Source ULP address 117 is neither of number of objects or content of objects.

Although element 118 is data length, element 118 is neither a number of objects nor a content of any object.

Lastly, although element 119 is the actual payload element data, the data itself cannot be either a number of objects or a content of any object.

Blocks 117, 118, and 119 have also been offered as disclosing iii) a number of field mappings and a content of any field mapping, any field mapping being usable by predetermined fields. Blocks 117, 118, and 119 disclose neither of a number of field mappings nor contents of any field mapping.

For the final part of the recited message content, iv) a number of actions and a content of any actions, any action being usable by the predetermined fields, the Official Action offered blocks 116 and 124.

As discussed above, element 116 indicates the number of payload elements (triplets 117-119). These are data payloads not "a number of actions". Block 124 is ULP message type field 124 indicating message type, message type not being an action. Also

block 124 is not part of the message content portion as it is outside the payload 129.

In summary, SAMUEL does not even disclose how payload is structured; rather SAMUEL only describes the size of the payload followed by the payload data itself.

Claim 28 has been cancelled without prejudice and may be the subject of a timely filed divisional application.

New claim 60 is believed to be patentable in view of the above discussion.

More specifically, the applied references taken either individually or in any reasonable combination are not seen as disclosing point-to-point communications between a sender and a receiver with messages of the recited flexible message format.

The prior art is not believed to teach or suggest messages comprising the header portion and the message content portion recited by claim 60.

The prior art is not believed to teach messages having a header portion comprising message definition references (MSG ID, MSG CLASS, MSG VERSION, MSG CREATOR), a sender identifier (SENDER ID) and a destination address (DESTINATION ADDRESS).

The prior art is also not believed to teach messages having the recited message content portion. The applied references were not found to teach or suggest a message content portion supporting each of:

i) a field count field (FIELD COUNT) and a first number of data fields (FIELD(1),...), the field count field storing the first number, and the data fields each storing message data,

ii) a object count field (OBJECT COUNT) and a second number of object fields (OBJECT(1),...), the object count field storing the second number, and the object fields each storing an object, each stored object being referred to one of the data fields,

iii) a map count field (MAP COUNT) and a third number of map fields (MAP(1),...), the map count field storing the third number, and the map fields each storing database mapping instruction data, and

iv) an action count field (ACTION COUNT) and a fourth number of action fields, the action count field storing the fourth number, and the action fields each storing an action command found in a message action list.

In addition to not teaching the recited message format, the prior art was not found to teach or suggest the recited point-to-point communications by interpreting and processing the messages using a database (ILMDB) storing a message definition table (msgdef), a field definition table (flddef), mapping instructions (fldmap) and message action lists (fldact, msgpre, msgpost).

Independent claims 37 and 56 are believed to be patentable for the reasons listed above.

The dependent claims are believed patentable at least for depending from an allowable independent claim. Also, the features recited by the claims are not all seen as being taught by the prior art. See, e.g., new claim 61 reciting the method of claim 60, wherein the sender identifier and the destination address are each email addresses in the form of a user name and a domain name (user-xyz@domain.com).

Applicants therefore believe that the present application is in condition for allowance and an early indication of the same is respectfully requested. Reconsideration and allowance of all the claims are respectfully requested.

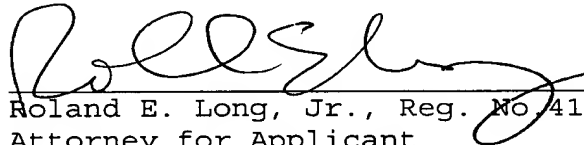
Should there be any matters that need to be resolved in the present application, the Examiner is respectfully requested to contact the undersigned at the telephone number listed below.

Please charge the fee of \$36 for the four extra claims of any type added to Deposit Account No. 25-0120.

The Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No. 25-0120 for any additional fees required under 37 C.F.R. § 1.16 or under 37 C.F.R. § 1.17.

Respectfully submitted,

YOUNG & THOMPSON



Roland E. Long, Jr., Reg. No. 41,949  
Attorney for Applicant  
745 South 23<sup>rd</sup> Street  
Arlington, VA 22202  
Telephone (703) 521-2297

REL:lrs

APPENDIX:

The Appendix includes the following items:

- substitute specification and marked-up copy
- a new set of formal drawings

**MARKED-UP COPY OF SPECIFICATION**Title

5           Method     of     object     oriented     point-to-point  
          communication   and   communication   apparatus   for  
          carrying out such a method

Field of the Invention

          The    present    invention    is    in    the    areas    of  
10    electronic data communications, database organized, flexible  
      format electronic data message management, client-server and  
      distributed software applications.

          In    particular,    this    invention    uses    a    relational  
      database to describe and manage the creation, communication,  
15    interpretation, display and real-time responses to flexible  
      format electronic data messages and applications organized  
      as collections of flexible format electronic data messages  
      which may be optionally distributed across one or more  
      computer systems on one or more electronic data networks.

20

Background of the Invention

          General purpose and business specific electronic  
      data communication meets with four distinct problems, each  
      of which has many sub-problems. We will be discussing these  
25    issues with regard to business communication needs, although



the following points can apply to almost any category of electronic data communications. The four problems are:

1) The electronic data that is sent and received must be clearly recognized for what it is meant to be. Each  
5 company has a different way of representing even the most basic business information, an order form or bill of lading, for example.

2) Once electronic data is received from another company and correctly recognized, it must be integrated into  
10 the receiving company's databases and business processing programs. Here again, each company has many differences in their database structures, which means the way they represent and store their electronic data on their computer systems.

15 3) If the electronic data must be displayed to an end user and allow for end user interaction, here again, each company may be using different types of computer display terminals with various types of display software, various system requirements and business requirements. It  
20 is very often necessary to build very expensive electronic data display programs with customized graphical user interfaces.

4) There may ~~exist~~be the need to execute the  
processing of computerized business functions in a  
25 coordinated manner, distributed across more than one

computer system, due to the need to use a combination of specialized system services and their associated data to accomplish specific business functions. This means that a computerized business system might need to exist in more  
5 than one location, operate differently in one or more locations, but function, in essence, as if it were one, single system. The solutions to this problem are usually classified as distributed applications or, more recently, distributed object management systems that are classified as  
10 "middleware".

There are really two distinct primary issues associated with problem four. The first issue is the need or desire of a company to use the same business application on many different systems, avoiding the need to rewrite the  
15 business logic, user interface and network communication interface for each different system that it needs to run on. The second issue is the need to divide the functionality of a single application over more than one system across a communications network, and have it operate in a fully  
20 coordinated manner.

There is no single system currently that can solve all of these four problems.

Businesses waste millions of dollars each year, buying, building, rebuilding and maintaining electronic data  
25 communication and storage systems that inefficiently and in

completely solve various combinations of these four problems.

#### Prior Art

5

#### Problem 1

To solve the first problem, a set of electronic data formats for particular types of information must be agreed upon by all communicating companies before the communication actually takes place. Furthermore, the  
10 agreement upon the formats must be actualized by the use of communication software that uses these same exact agreed upon formats. EDI (Electronic Data Interchange) has been one approach to solving this problem. EDI is a set of electronic data message formats for many types of business  
15 transactions. The formats are not flexible, meaning they cannot be changed by the users, and there are quite a few different versions of EDI messages. A company could implement an EDI message business communication system and still not be able to exchange electronic data with other  
20 companies using a different version EDI message set. The amount of time and money it takes to convert an existing business system to EDI is enormous and it is then difficult and expensive to change to, or add another format if that becomes necessary.

Methods and solutions using hard-coded (i. e. defined in the software) message formats have all of the problems associated with EDI and in essence are the same solution, only with a different fixed message format. Every  
5 time that a fixed message format must be changed, this requires a similar change in the software as well as possibly the database and user interface. Even a very small change can sometimes take months. And, of course, this electronic message format must be changed on the computers  
10 of every business using that particular format to insure that future electronic communication will continue correctly.

Essentially there will always be the requirement that, in some way, a common data format must be agreed upon  
15 by the sender and receiver. This will be present in any method available, now or in the future. The critical issues for the customer are: how fast and cheap it is to build a system around a method (i.e.: EDI) and how easy it is to change a previously agreed upon electronic data format, in  
20 every part of the system (database, software, user interface).

### Problem 2

The second problem is one that is harder to solve.  
25 In general, most companies use large numbers of data entry

personnel, who are reading printed material and entering the data that they have read into a computer program running on their terminal that will process this data, convert it to the correct format and place it in the company database.

5 Those companies using EDI or a similar common electronic message format will quite often have paid a very large amount of money for customized software that processes and converts the common electronic message format automatically for them and interacts with their existing database  
10 structure for the purpose of data storage and retrieval. If they need to change or add to the common message format, or they need to change or expand their database structure then they must spend time and much money to have this electronic message format database interface program changed and re-  
15 written again.

One newer public domain general purpose solution centers around on "mapping" specific data items to specific locations in a customer database or storage device. This mapping is not created in software, but is in some more  
20 easily changeable and flexible representation that is separate from the software application and customer database, such as an independent flat file that the software application reads. This eliminates the need to reprogram and rebuild the software application every time the database  
25 structure or the data representation changes. Data field

mapping techniques are in common use by many database companies such as Progress, Oracle and SQLBase (SQL = Structured Query Language). A version of this well known method is part of the functionality and methodology  
5 incorporated into communication apparatus of the present invention.

### Problem 3

The third problem has had few easily workable  
10 general purpose solutions. One category of solutions requires writing extensive amounts of software code for various window (graphical user interface) management systems such as X-Windows or Microsoft Windows. For large, multi-functional business applications, this approach requires a  
15 tremendous amount of time, money and effort, and makes changes or additions to the user interface a very time consuming and costly affair as well.

A new and promising category of solutions centers around the use of HTML (Hyper Text Markup Language) and web  
20 browsers, which can dynamically create very nice looking graphical user interfaces, usually called 'web pages', because they are used on the World Wide Web and the internet. This has aroused the interest of many companies. They see how quickly and cheaply very nice looking graphical  
25 user interfaces can be built using HTML and they wonder if

the user interfaces for their electronic data communication and processing programs could be created in the same way and communicate over the same networks as the web browser systems. In fact, companies would like to be able to use the  
5 internet to send and receive the full range of business and personal data and messages.

At the present time there are several systems available that will allow a limited range of business and general data communication over the internet, using HTML and  
10 web browsers. The reason that the types of data communication are limited is that the webserver/web browser systems were not designed to provide full data communication services. They were designed as dynamic document retrieval and display systems. They can retrieve information that is  
15 first initiated as a request from the user. They do not give the user the capability to send information to one or more destinations, except by extensive additional programming. The capability for sending is not inherent in the system. Examples of this type are Progress' WebSpeed, IBM's WebEDI  
20 and SapphireGroup's PageBlazer.

Even the "newer" features of web browsers and web servers that implement the so-called "push" technology only allow the receipt of data, not the sending. And this "receipt" is actually set up as a simulated broadcast  
25 receiver, a hidden repeating request for information

(polling), by the web browser and webserver systems. The plans to expand this "push" technology only include the use of true broadcast and multi-cast communications technology. Examples of "push" technology are Marimba's Castanet and Tibco's TIB API. This so-called "push" technology is of great interest to advertisers that wish to reach a mass market in a new way, but is of almost no use to the company or individual wishing or needing to use the advanced document display features of HTML and the Internet in the context of fully integrated two way communication of information using point-to-point protocols that can be easily integrated into their existing database and business systems.

The webserver and web browser interfaces for the Internet are set up to allow end users to send messages as requests for web pages. At no time will a web browser user be able to send a message to another web browser user, or send a web page to another web browser user, except through electronic mail. At no time will a web browser user ever receive a specific message that was not first asked for; even the so called "push" technology information is requested by polling and the nature of the information, even when it is customized can be compared to a television broadcast or a mass mailing. The information is not being sent to one specific recipient. Contrast this with



electronic mail (email), for instance. Using email services one can receive any number of specific, private, messages without asking the sender for them first. One can send any number of specific, private messages to particular  
5 recipients, without first being asked. This is the ordinary and necessary nature of messages. Yet the webserver systems of the internet were not designed to communicate like this.

Why then are companies and people so interested in using them this way? Why don't they just use email? The  
10 answer is that email does not have any data formatting services built into it (as HTML does) and does not have any user interface generation system built into it (as web browsers do). An email message may also take an indeterminate amount of time (several seconds to several  
15 days) to reach its destination and sometimes can get lost. This is unacceptable for standard business electronic data communications. That is why companies and people like the webserver/web browser system, which has a very good data display capability built into it and moderately fast  
20 communications. However, as mentioned above it does not have point-to-point communication capability or data representation and mapping capability.

In the present invention the public domain HTML standard may be incorporated as the solution for generating

very easily built, fast and flexible graphical user interfaces.

#### Problem 4

5 Historically, problem four has been addressed in a general way, until very recently, only by Relational Database Management Systems (RDBMS). This class of solutions has been and is restricted to solving the proper functionality of distributed query and distributed database  
10 management and does not address the primary two issues of problem four as described earlier.

A second and newer class of solutions ~~are~~is to be found in systems currently referred to as "middleware" or distributed object management systems. A good example of  
15 this is a system called CORBA (Common Object Request Broker Architecture). An excellent set of articles on the development history of CORBA, the goals, current functionality and deficiencies of CORBA 2.0 can be found in Warren Kayffill, "CORBA masterminds object management", p.  
20 42, DBMS magazine, volume 10, number 3, March 1997, and T. J. Hart "Questioning CORBA. Bringing Corba-based designs to life faces a multitude of obstacles", p. 52 in the same issue of DBMS.

This class of solution does not address the  
25 database interoperability issues, the common data format

issues at the application level or user interface issues. It addresses solely the ability to write an application with standard communication interfaces and standard data at the communication level. Essentially this type of system  
5 supplies a development environment, a run-time environment and set of software libraries that allow programmers to create applications that will run on any system and, after defining data structures in the code (so-called objects), these data structures can be exchanged with other software  
10 programs that have been coded to recognize and use those particular data structures.

This capability and functionality is, in essence, not any different than a programmer writing a message communication system in Java, which runs on almost any  
15 system, and using either EDI messages or some other agreed upon fixed format as a communication medium. The CORBA system does not interpret any messages. The application, written by the programmer, must be able to recognize the message. This leaves a CORBA application with all of the  
20 same issues for cost and time of development, alteration and maintenance as an EDI or fixed message format application system. Even the distributed functionality that is possible with CORBA is not inherent in CORBA, but in the skill of the programmer designing and coding the application and use of  
25 the CORBA communication libraries. In essence, the only

apparent gain of CORBA is the interoperability of part of an application system across many run-time environments, which can perhaps be better gained through other means, such as Java.

5           In essence, it appears that "middleware" in most cases is in fact just another electronic data communication environment that offers a standardized communication environment and some standardized software functionality on many computers, but fails to address all of the aspects of  
10 software application building and so requires the use of system dependent code for database interfaces and user interfaces, thereby defeating the main purpose for the use of a system such as CORBA.

          There are some systems that are merging CORBA  
15 functionality into Java and also including a standardized database interface called ODBC, or in the Java system JDBC. This is an interesting development, but as the ODBC database interface is only efficient for read-only queries, standard business transaction systems, which require full database  
20 access, can not be successfully created. Also, this combination does not address problems one or two at all. In addition, it is not the desired platform on which to build the present invention due to the use of ODBC and the unwieldy overhead of the CORBA object broker system.

Further prior art is known from some patent documents which will be briefly discussed below. US 5,257,369 (Tibco, Inc) discloses "middleware" presented earlier.

5 In addition, this document provides broadcast functionality and is based upon broadcast functionality. It is primarily to provide a solution to the problem of delivering realtime data to many recipients at once, such as in real-time stock quotes or market information, and does  
10 not address point-to-point communications or problems 2 or 3 at all.

The message format is able to be changed and also defined as a data structure or object as described in CORBA earlier. However, this capability does not address the  
15 ability to change or add a message format without changing application code, or the ability to map the data of a message format to either a user interface or database fields.

WO 96/20553 (Alphanet Telecom, Inc) focuses upon  
20 the enhancement of electronic mail (email) services to include voice mail and facsimile (fax) information. The description of the system is of a centralized service which users can connect to for the receipt of voice mail, email or fax through the Internet, phone or fax machine connecting to  
25 the same service. This invention does not address in any way

problems 1,2,3, or 4, but does address a way of possibly making email more efficient from a communications point of view.

WO 96/34341 (Charles Bobo II) focuses upon a  
5 centralized service, that collects, stores and allows access to data for connected users. This is entirely different from the point-to-point communications necessary to solve problems 1 through 4 and for most types of standard business communications. In addition, there are many commercially  
10 available systems that have been in existence for many years that already function in exactly this way, such as GEIS and the IBM private data network, to name only two.

EP-0,747,840 A1, EP-0,747,841 A1, EP-0,747,842 A1, EP-0,747,843 A1, EP-0,747,844 A1, and EP-0,747,845 A1 (all  
15 of International Business Machines Corporation) are focused upon enhancing the current webserver functionality with some forms of database access and so are therefore still bound by the communications limitations of webserver discussed earlier. Therefore this set of inventions can not solve  
20 problems 1 and 4 at all and 2 and 3 in only a limited way. In addition, there are several commercially available systems that appear to offer the same or even more functionality in webserver enhancement such as WebSpeed from Progress Database Corporation.

WO 95/11560 (Sybase, Inc.) is in the category of "middleware" that solves only the problem of creating a consistent electronic data communications software interface which applications may be built upon, regardless of the computer hardware or operating system being used. This document does not address the format of the electronic data being communicated in any manner and so this solution is very similar to CORBA and has all of the same limitations from the point of view of the four problems being addressed by the present invention.

EP 0 421 624 A2 (Texas Instruments Incorporated) is a combination of a "middleware" system such as CORBA, and an application development environment.

Using a database controlled and administered software development environment, the claimed invention allows programmers to develop applications in C, COBOL and SQL.

The database contains information which allows large groups of programmers to develop, modify and use source code control techniques for complex database transaction applications. The database also contains information that allows the application to be recompiled on alternate hardware systems and function in the same manner on each type of computer, although in this document mostly IBM mainframe systems and some Unix systems are discussed.

The end result is a series of hard-coded, compiled applications, with all of the same issues as described in problems 1 and 2 and only partially solving problems 3 and 4.

5                   EP-0,456,249 A2 (Hewlett-Packard Company) is a combination of a "middleware" system such as CORBA, and a partial application development environment whose primary capability is linking existing applications that have been developed in differing higher level programming languages.

10 The higher level languages that are cited in this invention have different run-time in-memory organizations of their data and must transform the data from the organization of one language to another. To this end, an intermediate data representation and data transfer commands are defined that

15 enable the invention, after compiling the data representation and transfer commands, to translate the identified data structures from the runtime representation of one language to another. This invention does not address in any way a message format that contains all of the data

20 and commands necessary to run independently as an application, only the data and commands necessary to translate specified data structures between two pre-existing applications. Therefore, this invention does not address the issues discussed in problems 2 and 3 at all and only

25 partially addresses the issues in problems 1 and 4 at the



level of the programming language, not at the level of the application as it has been discussed here. In contrast, in the current invention, the applications are composed entirely of the messages and therefore there is no translation of data from one language to another and no compilation of message formats or commands required; all of the data remains in the message format, whether it is being transmitted or it resides in the computer memory.

Given the above, an urgent need arises for a single system and a way of communicating that can solve each of these four issues and combine the solutions into one unified whole. This would be of great benefit to the individuals and companies that must build complex electronic data communication systems and who must face these issues on a daily basis. Such a system and communication method would reduce the time, cost and complexity of building and maintaining electronic data communication systems drastically.

#### Summary of the Invention

To obtain the object referred to above, the present invention ~~claims~~ is a method of point-to-point communication between a sender and a receiver by means of messages with flexible message formats, characterized in that any of the messages comprises:

-a header at least comprising message definition references, a sender identifier and a destination address;

-message content regarding:

- \* number of fields and content of any field,
- 5       \* number of objects and content of any object,
- \* number of field mappings and content of any field mapping, any field mapping being usable by predetermined fields;
- \* number of actions and content of any actions,
- 10     any action being at least usable by predetermined fields.

Consequently, all of the elements for defining application level functionality are present in the message format, which is already in the context of a network communication system and therefore may be easily distributed  
15 by simply changing a location and destination address.

Moreover, in accordance with the present invention, a communication apparatus is claimed which comprises processing means and a database, arranged for point-to-point communication with another communication  
20 apparatus by means of messages with flexible message formats, the messages comprising:

-a header at least comprising message definition references, a sender identifier and a destination address;

-message content regarding:

- 25       \* number of fields and content of any field,

\* number of objects and content of any object,  
\* number of field mappings and content of any field mapping, any field mapping being usable by predetermined fields;  
5       \* number of actions and content of any actions, any action being at least usable by predetermined fields;  
the processing means being arranged for consulting a predetermined message definition table stored in the database using the message definition references as  
10 references to the predetermined message definition table.  
Thus, in the present invention a flexible message format is defined that can contain any type of data and that contains enough information to fully describe the data within the message itself. Any message comprises message  
15 definition references which are used by the claimed communication apparatus receiving the message to identify a message definition preloaded in its database. However, the technology is flexible since the preloaded message definition is not fixed for once and for all. Within the  
20 communication apparatus receiving the message, message format specific instances can be created, edited or deleted by inserting, updating or deleting entries in relational database tables which fully describe the message and reference it with a unique identifier. Moreover, if message  
25 definitions are absent at a desired location, entire

database message definitions may be exchanged automatically for particular messages or groups of messages, thereby creating easily exchanged common format messages for multiple users. A further advantage of the present invention  
5 is the small amount of necessary overhead.

Examples of fields within the message definition references are a message identifier for identifying any of the messages, a message class identifier for identifying a message class for any of the messages, like mail, business  
10 message, orders or shipping, a message version identifier for identifying a version number of any of the messages, and a message creator identifier for identifying a creator of any of the messages.

Any of these fields within the message definition  
15 references have equivalent counterparts within the message definition table in the claimed communication apparatus.

Moreover, headers of messages may include a reference to a type of encryption and/or compression used. Also these references, then, have counterparts in the  
20 message definition table in the communication apparatus.

Preferably, the content of messages is protected by digital signatures.

Preferably, the database of the communication apparatus is organized in several tables to which reference  
25 is made from the message definition table. Then, the

predetermined message definition table comprises a message system identifier for use as a reference to further tables in the database.

One of such further tables may be a field  
5 definition table for holding primary definitions for any field of the messages.

Preferably, one of such further tables is a field mapping table comprising mapping information usable by predetermined fields, e. g. for mappings to hyper text  
10 markup language fields, database fields, flat file fields and other message fields, said database fields and flat file fields being stored in a customer database. Then, the message format according to the invention contains not only data, but also mapping commands for user interfaces,  
15 customer database fields, and sequences of actions including calculations and conditional logic. Consequently, all of the elements for defining application level functionality are present in the message format, which is already in the context of a network communication system and therefore may  
20 be easily distributed by simply changing a location and destination address. Moreover, in this way the HTML standard may be incorporated for generating very easily built, fast and flexible graphical user interfaces, allowing a direct mapping to and from the message format defined by the  
25 invention, as managed, defined and controlled by the claimed

communication apparatus and displayed and interacted with on a user terminal, like a microprocessor, connected to the communication apparatus.

Additionally, one of such further tables may be a  
5 field action table comprising the message action links usable by predetermined fields.

Preferably, further tables are a message pre-processing table comprising a list of actions to be executed as pre-processing for a message either received or to be  
10 sent and a message post-processing table comprising a list of actions to be executed as post-processing for a message received.

The field action table, the message pre-processing table and the message post-processing table comprise  
15 references to types of action selected from the following group of actions: database type of actions and logical type of actions including mathematical calculations, assignments, logical operations, conditional operations, and commands.

Preferably, in the present invention, applications  
20 are defined as a named group of one or more messages. To that end, the message definition table comprises an application field for indicating whether a message received is a first message of an application and an application name field for referring to a name of the application.

In accordance with the present invention, the claimed communication apparatus is, preferably, arranged for requesting a new message definition from a sender if a message received refers to a message definition not present  
5 in its database, and receiving said new message definition from said sender and storing it in said message definition table in said database.

The processing means of the communication apparatus may be arranged to either merge a message received  
10 with a designated HTML file or if the designated HTML file is not found by the processing means, to create a default dynamic HTML file. However, alternatively, the processing means of the communication apparatus may instruct a connected terminal to carry out these functions. Therefore,  
15 the present invention also relates to a system comprising a communication apparatus as defined above and a terminal connected to the communication apparatus, the terminal comprising a terminal processor, a display unit and input means for inputting data by a user, the communication  
20 apparatus being arranged for passing a message received to the terminal if the terminal is indicated in the message to be the destination address, and the terminal processor is arranged to either merge the message with a designated HTML file or if the designated HTML file is not found by the  
25 terminal processor, to create a default dynamic HTML.

Thus, user terminals that receive a message for which the terminal processor does not have the corresponding message definition can rely upon the system's automatic interpretation and display which is based upon the self-describing information contained within the message received. This same self-describing information can then be used by the terminal processor to automatically create a default database message definition if the database message definition referred to by the message can not be found.

Such a default database message definition can then be edited and adapted to the local system by the terminal user.

The present invention may include a state of the art solution for problems two and three combined in a unique way with completely new solutions to problems one and four, all built around a unique flexible format electronic data message management system composed of the claimed flexible message format and communication apparatus.

The message management system of the present invention, which is defined and controlled by detailed information stored in the tables of a relational database, allows for the most efficient, complete and simple to use way of communicating electronic data in flexible formats. The flexible message format allows the inclusion of any form of electronic data. The flexible message format contains



information that allows a receiver to interpret, understand and display the contents of the message, even if it has previously never been received. The receiver can also add this unknown message format to its own message database and  
5 link it into its business databases and an HTML user interface very easily and quickly, without writing or changing any software. An HTML user interface can also be dynamically generated by the software of any terminal in the case where there is no specified, previously created HTML  
10 file. This addresses the first issue.

In the present invention, an ANSI standard SQL interface to the major relational databases may be provided. This interface provides the service of automatically linking the user selected fields of a data message to the fields in  
15 a relational database table, for subsequent storage or retrieval. This data mapping interface can also link to flat file formats such as fixed length fields and CSV (comma separated value) files.

This data mapping interface can be expanded to  
20 link message data fields to many other formats, such as SNMP and manufacturing protocols for automatic process control.

This data mapping interface is defined by entries in the ILMDB and controlled by the ILMS. This addresses the second issue.

In the present invention, a message control user interface terminal may be provided, where a user may create, edit, receive, send and view messages. The message control user interface communicates to a network and message management server arranged in accordance with the invention. In essence, the present invention replaces the current webserver/web browser technology for those individuals or companies that need full data communications capability, and expands the system domain to include any data communications network. This allows the full sending and receiving of HTML, VRML and Java appletes, as well as any specific data the user has defined to be in a particular message. When combined with the previously described features, this provides the full, database controlled linking between the data in a message, the customer data source it must be retrieved from or stored in, and the graphical user interface (HTML, etc.) it must be displayed with.

In addition to solving the issues described above, the present invention has added the capability of defining automatic, system performed actions, that may be organized as message pre and post processing, or, in the case of messages linked with a graphical user interface, field selection actions. These actions may be defined as SQL or as an "InterLingua<sup>TM</sup> Script". "InterLingua<sup>TM</sup> Script" (IS) is defined in the present invention and is a simple, yet

powerful set of calculation, execution and conditional logic control statements, that are dynamically interpreted and executed in real-time, and can interact with all message data fields. The actions can generate a new message, call  
5 built in system functions and external programs or send and receive database queries. These actions can be used to generate an automatic message response, automatic field validation, perform calculations, run a Java applete or a complex database transaction, for example.

10 In the present invention we have included the truly unique method of an application being defined as a set of messages and their associated actions. Due to the richness and flexibility of the interaction between the messages; their ability to run database transactions, merge  
15 with or generate graphical user interfaces, or run silently in the system as a background process, it is possible to fully define a complex application as a collection of data messages and their associated actions. This also has vast implications in the realm of distributed application  
20 processing, as these messages are, by their very nature, sent, received, understood and organized with great ease, as a basic function of the system. This means that for the first time, a distributed application system could easily and flexibly define, without changing or adding any software  
25 code, the local user interface and customer data source

interface (such as the business database) at each site and still be using the same application (message set).

#### Brief Description of the Drawings

5           The invention will be explained with reference to some drawings which are only intended to illustrate and not to limit the scope of the invention.

Figure 1 shows a general overview in block diagram of the system according to the invention;

10           Figures 2a, 2b, 2c, 2d, and 2e show a message format according to the invention.

#### Description of the Preferred Embodiment

In figure 1 a plurality of servers  $SRV(m)$ ,  $m = 1, \dots, M$ , is shown. Any of the servers  $SRV(m)$  may be  
15 connected to one or more terminals. The terminals connected to server  $SRV(1)$  are referred to with  $ILMC(n)$ ,  $n = 1, \dots, N$ . The acronym "ILMC" has been derived from InterLingua™ Message Client. The connections are numbered with  
20 references 4, 5.

Moreover, server  $SRV(1)$  is connected to a customer database  $CDB(1)$ . Although one customer database  $CDB(1)$  is shown, more than one can be applied as required.

Terminal  $ILMC(1)$  has been depicted in greater  
25 detail in figure 1. Terminal  $ILMC(1)$  is shown to include a

monitor screen 6, a processing unit 1, a keyboard 12, and a mouse 13. The keyboard 12 and the mouse 13 are connected to the processing unit 1. Of course, any other known means for inputting data to the processing unit 1 by a user is possible instead of or in addition to the keyboard 12 and the mouse 13.

The monitor screen 6 shows some "buttons" 7, 8, 9, 10, and 11. These buttons can be acted upon by the user by means of the mouse 13 to instruct the processing unit 1 to carry out predetermined tasks. Such predetermined tasks may be the opening of options, establishing connections, sending messages, receiving messages, and/or loading of data into the memory of the processing unit 1.

The server SRV(1) is provided with a processor called InterLingua™ message server ILMS for receiving, passing or sending messages in accordance with the method of the present invention. Part of the ILMS processor is indicated with IS (= InterLingua™ Script), the function of which will be explained later.

Server SRV(1) also comprises a database ILMDB (= InterLingua™ Message Database) for storing several tables in accordance with the present invention, as will be explained later.

The server SRV(1) is arranged for point-to-point communication with one other server through a communication

path 2. The communication path 2 may be established in any known way, i.e., either by wires or wireless.

Figure 1 shows several other servers SRV(m) connected to one or more terminals ILMC's. These further  
5 servers SRV(m) are organized in accordance with the present invention further explained in detail below.

Figures 2a, 2b, 2c, 2d, and 2e show a message format. In figure 2a, the message format is shown to include four fields MSG Id, MSG Class, MSG Version and MSG Creator,  
10 used as message definition references by any apparatus receiving the message.

The field MSG Id is to identify the message.

The MSG Class is used to designate a category or type of message, like mail, business message, orders, or  
15 shipping.

The field MSG VERSION comprises a message version identifier for identifying a version number of the message concerned.

The fourth field MSG CREATOR comprises a message  
20 creator identifier for identifying a creator of the message concerned.

The MSG Id, Class and Version fields are integer values. There are, e.g., 1,000,000 MSG Ids per MSG Class and 10,000 MSG Classes. Every MSG Id, MSG Class pair can have,  
25 e.g., 100 MSG Versions. MSG Classes may e.g. be arranged as

follows. MSG Classes 1 to 20 are reserved for the ILMS. MSG  
Classes 21-100 are pre-defined categories such as order  
placement, shipping, etc. The 9900 remaining MSG Classes are  
for user definition. The MSG Creator may e.g. have 80  
5 characters.

This is for the purpose of identifying entire  
message sets created by particular users or companies for  
easy automatic transfer.

Together with five more fields these message  
10 definition references form a header to the message. These  
five more fields are: SENDER Id, DESTINATION ADDRESS,  
ENCRYPTION TYPE, COMPRESSION TYPE, and ~~AND~~ APPLICATION NAME.

The Sender Id is the user name as recognized by  
the server SRV(m) with the name of the server SRV(m)  
15 appended, e.g., in a manner similar to an email address  
(i.e. user-xyz@ilserver.com). The field DESTINATION ADDRESS  
is for identifying the address to which the message  
concerned is to be transmitted. The destination address may  
be of the same format as the SENDER Id.

20 The field ENCRYPTION TYPE is used for a reference  
to a type of encryption applied, if one is required. The  
field COMPRESSION TYPE is to indicate the type of  
compression used, if required.

The field APPLICATION NAME is used for indicating whether or not the message concerned is member of a series of messages which together form one application.

The Field Count gives the number of fields, e.g.,  
5 from 0 to 256 in the message. A "field" and its content refer to a particular type of data and the data itself. The actual field data follows this field, if the count is more than 0. With reference to figure 2b, a field is defined by a  
10 Type identifier, a Size value, a field Name for internal identification, a field Label for default display, and a field Description, also for default display. The length of data following is known~~knew~~ by combining the information of the Type and Size fields. One field definition follows directly after another with no additional separators.

15 Very often, particularly in HTML type data, there is a reference to a local file containing binary image, sound or program data. To insure the correct transfer and referencing of these referenced external objects, they are automatically grouped at the end of the message, as shown in  
20 figure 2a. The fields that reference them are made to point to the objects. When the message is received, these objects are automatically stored locally and the field references are automatically adjusted to access the objects in their new location.



The Object Count field gives the number of objects in the message from 0 to 256. The object data follows this field, if the count is more than 0. An Object is, preferably, defined by a MIME Type/Content identifier, a  
5 Size value and then the actual Object Data, as shown in figure 2c. One Object definition follows directly after another, with no additional separators.

With reference to figure 2d, a field mapping is e.g. defined by a FIELD IDENTIFIER, a MAPPING TYPE, and  
10 MAPPING DATA.

As Figure 2e shows, actions can be simply defined by ACTION TYPE followed by ACTION DATA.

A digital signature, using a server SRV key, is, preferably, created on the entire message and added at the  
15 end of the Object list.

The allowed data types as defined in figures 2a, 2b, 2c, 2d, and 2e are for instance:

- 1) Unicode character strings;
- 2) Computer hardware independent integers of any  
20 size;
- 3) IEEE 8-byte format floating point numbers;
- 4) Universal Resource Locators (URL);
- 5) Image data in GIF or JPEG format; and
- 6) MIME specified data representations.

In actuality, GIF and JPEG data are also included as MIME types. The decision to make a specific data type for images in these formats was driven by the frequency of their use in HTML types of display systems. The most commonly used data types have their own definition. Any additional data types can be included as existing MIME types or specific MIME extensions. These data types are specified by way of example only. The scope of protection of the present is not intended to be limited by the type of data preferred.

Each message may have its own digital signature and optional encryption and/or compression, which is managed automatically by the server SRV(m).

#### Message Database

A flexible message format definition, management and control system is implemented using database ILMDB as a relational database as a holder of

- a) detailed message definitions;
- b) message mapping instructions;
  - b.1) mapping to customer database fields;
  - b.2) mapping to customer flat file formats;
  - b.3) mapping to customer user interface (HTML) fields; and
  - b. 4) mapping to other message fields;

c) message action lists: (SQL, Java, calculations,  
logic flow, message calls, external program calls)

c.1) message pre-processing actions;

c.2) message post-processing actions;

5 c.3) message field user interface event  
actions.

This information is held in tables in the  
relational database ILMDB and allows the system to fully  
interpret and process any message. New message definitions  
10 may be added and old definitions may be changed. Message  
definitions may be sent from one server SRV(m) to another  
for immediate incorporation into the message database ILMDB.  
Every flexible message format includes enough self-  
description information in itself to allow a server SRV(m)  
15 to interpret a previously unseen message and create a new  
message definition entry in the ILMDB for it automatically.  
This new message definition may be edited and enhanced by a  
system administrator, to complete the mapping into the  
receiver's system.

20 Message data is not kept in the database ILMDB,  
only the definition. The message data is kept in compressed  
files on a disk storage system. These files are periodically  
either archived or deleted according to the server message  
duration options chosen by the system administrator.

The primary relational tables within the database ILMDB needed to define and control a message either received or to be sent are as follows:

	<u>Table Name</u>	<u>Description</u>
5	msgdef	InterLingua <sup>™</sup> Message Definition
	flddef	Message Field Definition
	fldmap	Field Mapping
	fldact	Field Action Definition (one action per field)
10	msgpre	Message Pre-processing (sequence of message actions)
	msgpost	Message Post-processing (sequence of message actions)

15           System developers skilled in the art will recognize that there are many other tables of related information necessary to make an entire system more easily usable, and control secondary system features. Examples of this might be all of the tables associated with user  
20 accounts and user information, system administration and system security. The tables presented here are the core tables necessary for implementing the present invention in accordance with the preferred embodiment, and extensions or enhancements are considered to be within the scope of the  
25 present invention as well.

Table:msgdef

This table is responsible for holding the primary definition of a message. A message is completely identified by the first four fields of this table, which are also in the message header (cf. figure 2a). The fifth field (msysid) is an identifier that is assigned internally by the server processor ILMS, and is used as a reference for identification of and fast access to other tables within the database.

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msgid	integer	Message Identifier
	msgver	integer	Message Version
	msgclass	integer	Message Class
15	creatid	char	Creator Identifier
	msysid	integer	Message System Id
	creatdate	date	Date Created
	usedate	date	Date Used
	These two fields control how long a message is		
20	stored by the ILMS.		
	durdays	integer	Duration Days
	durmins	integer	Duration Minutes
	These two fields identify the type of encryption		
	and digital signature algorithm.		
25	encrtype	char	Encryption Type

sigtype char Digital Signature Type

This field indicates whether the message is for display or not.

silent boolean Message Silent Flag

5 This field indicates whether the message can initiate or respond to actions.

active boolean Processing Enable Flag

msgicon char Message Icon File (for ILMC display)

10 msghtml char Message HTML File (for ILMC display)

This field indicates whether the message is the first in an application.

appmain boolean Application Main Flag

15 This field contains a name if the message is part of an application.

appname char Application Name

descr char Message Description

20 Table: flddef

This table is responsible for holding the primary definition for all fields of all messages. A field is uniquely identified by the 'msysid' assigned in the 'msgdef' table and either a field sequence number (fldseq) or a field name (fldname).

25

<u>Field Name</u>	<u>Type</u>	<u>Description</u>
-------------------	-------------	--------------------

msysid	integer	Message System Id
--------	---------	-------------------

fldseq	integer	Field Sequence
--------	---------	----------------

fldname	char	Field Name
---------	------	------------

5            These three fields are used to create 1, 2  
(tables), or 3 dimensional arrays.

ar1	integer	Array Dimension 1
-----	---------	-------------------

ar2	integer	Array Dimension 2
-----	---------	-------------------

ar3	integer	Array Dimension 3
-----	---------	-------------------

10           fldtype           char           Field Data Type

The field size is interpreted differently,  
depending on the data type.

fldsize	integer	Field Size
---------	---------	------------

15           This field is used if the field is a reference to  
another message as a data structure.

smsysid	integer	Sub-Message System Id
---------	---------	-----------------------

These two fields are used if the field contains  
MIME data.

mimecont	char	MIME Content Identifier
----------	------	-------------------------

20           mimetype           char           MIME Type Identifier

fldlabel	char	Field Label (for ILMC display)
----------	------	-----------------------------------

fldcom	char	Field Comment (for ILMC display)
--------	------	-------------------------------------

25

### Table:fldmap

This table is responsible for holding any mapping information that a field might use. This table can define mappings to database fields, flat file fields and other message fields. The field is identified in the same manner as table 'flddef'. There is one mapping for data and one mapping for display allowed for each field, except in the case where a field is defined as an array. A data and display mapping can be assigned to one or more elements or dimensional rows of an array.

<u>Field Name</u>	<u>Type</u>	<u>Description</u>
msysid	integer	Message System Id
fldseq	integer	Field Sequence
fldname	char	Field Name

These three fields allow mappings to specific array elements or dimension rows.

ar1	integer	Array Dimension 1
ar2	integer	Array Dimension 2
ar3	integer	Array Dimension 3

This field allows the data to be transferred from the most recently active previous message within the same message context, which is the set of currently active messages for a particular user connection.

cpyfld	char	Copy Field (From Msg)
--------	------	-----------------------



These three fields are used to specify a particular field in a SQL accessible relational database.

dbnam	char	Database Name
tblnam	char	Table Name
5 fldnam	char	Field Name

This field specifies a field in an HTML file (msgdef.msghtml) for display.

tagnam	char	HTML Tag Name
--------	------	---------------

These two fields specify a flat file of either  
10 fixed field lengths, or comma separated values.

flatfile	char	Flat File Name
filetype	char	File Type (Fixed, CSV)

These four fields specify the field location for the flat file.

15 rowbegin	integer	Row Begin
colbegin	integer	Column Begin (or Field #
for CSV)		
rowend	integer	Row End
colend	integer	Column End

20

Table: fldact

This table is responsible for holding any action information that a field might use. Actions may be database type of actions and logical type of actions. There is one  
25 action per field, except in the case where a field is

defined as an array. An action can be assigned to one or more elements or dimensional rows of an array.

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
5	fldseq	integer	Field Sequence
	fldname	char	Field Name
	ar1	integer	Array Dimension 1
	ar2	integer	Array Dimension 2
	ar3	integer	Array Dimension 3
10	This field defines the action type : SQL or IS, file or local.		
	acttype	char	Action Type
	This field contains SQL, IS or a file name whose contents are SQL or IS.		
15	actscript	char	Action Script

#### Tables: msgpre and msgpost

The two tables msgpre and msgpost have an identical structure. Table 'msgpre' holds the list of  
20 actions to be executed by server processor ILMS as preprocessing for a particular message. This means before any display or field actions take place, the server processor ILMS completes the list or pre-process functions.

Table 'msgpost' holds the list of actions to be  
25 executed by the server processor ILMS as post-processing for

a particular message. This means that after all other message actions and displays have completed, the server processor ILMS completes the list of post-process functions before deleting the message from main memory. The process sequence number is used to create a controlled execution of the list of actions. A previous sequence number must complete before the next one can begin. If there is more than one action with the same sequence number, they are started at the same time, or, within a multiple CPU or massively parallel environment, as true parallel processes.

Table: msgpre

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
15	procseq	integer	Process Sequence
	This field defines the action type: SQL or IS, file or local.		
	proctype	char	Pre-processType
	This field contains SQL, IS or a file name whose contents are SQL or IS.		
20	procsript	char	Pre-process Script

Table: msgpost

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
25	procseq	integer	Processing Sequence

This field defines the action type: SQL or IS, file or local.

proctype            char            Post-processType

This field contains SQL, IS or a file name whose  
5 contents are SQL or IS.

procscript        char            Post-process Script

InterLingua™ Script (Action Definition and Control  
Language)

10            Any action, for either message pre-processing,  
post-processing or field actions can be either an SQL-action  
or IS-action. SQL (Structured Query Language) is an ANSI  
standard interface language for relational databases.  
IS(InterLingua™ Script) is a predetermined part of the  
15 server processor ILMS(cf. figure 1) and is a small, but  
powerful set of calculation, execution and conditional logic  
statements that give application level functionality to a  
message. In a preferred embodiment, the definition of IS is  
as follows:

20            Calculation

The operations of add (+), subtract (-),  
multiply(\*) and divide (/) are allowed upon integers and  
floating point values. Parenthesis "()" are allowed for  
grouping mathematical operations.

25

### Assignment

Assignment(=) is allowed to integers, floating point values, strings and URLs.

### Logical and Conditional

5            Symbols representing: and (AND), or (OR), greater than ( > ), less than ( < ), greater than or equal to ( > =), less than or equal to ( < =), equality (=), not equal (!=), and negation(!) are allowed upon integers, floating point values, strings and URLs.

### 10            Control Flow

          END-MSG- End a message. Must be last command from its own pre-processing list.

          END-APP- End an application. Must be last command from a member message post-processing list.

15            For Loop:

```
          FOR i = x TO y
          BEGIN
              < statements >
          END
```

20

          While Loop:

```
          WHILE < condition >
          BEGIN
              < statements >
```

25

```
          END
```

If-Then-Else:

```

    IF < condition > THEN
    BEGIN
5         < statements >
    END
    ELSE
    BEGIN
        < statements >
10    END
```

Command

Load mapped field data:

```
LOAD{WHERE < external field name > = < value > }
```

15 Store mapped field data:

```
STORE{WHERE < external field name > = < value > }
```

Execute external program:

```
CALL[JAVA | EXEC | SHELL] < program name >
20 < parameter 1 > ... < parameter n >
```

Execute message:

```
ILMP:// < host-name > / < message id >
```

Execute internal function (this is used mostly  
25 for user interface events):

ILMP:// < host-name > / < message id >/ < field id  
> [UI-ACT | MSG-END |APP-END]

ILMP:// < host-name > / < message id > / < field  
id > [ADD-DATA| DEL-DATA | CHG-DATA] < data >

5

### Applications

A collection of InterLingua™ messages may be grouped together, uniquely named and used as an InterLingua™ application. For the purposes of this discussion, for any programming language or system platform, an application is defined as being one or more functions whose sequence of operation may vary depending upon input data and user and system events; that can access and interact with external storage devices; and that can interact with users through a user interface.

All or part of this named set of messages may reside on one or more servers SRV(m) in one or more locations. This named set of messages, or application, implements:

- a) a directed series of functions
- b) local and shared data
- c) user interface screens
- d) networked distributed application functionality

The directed series of functions may include any of the action types, such as database queries, calculations

and the activation of additional messages. Activating a message in this context can be used for the equivalent of calling an application function, displaying the next user interface screen, or, of course, sending a message to a local or external network destination. The real-time structured flow of the order of operations, or, in other words, the directed series of functions, is determined by the message action lists and also event actions generated by the user interface, such as a mouse selection of a field or a button.

The local and shared data includes any of the allowed data types. These data types may be grouped into data structures. This message data may be mapped to and from user databases, user interfaces and passed to other messages. This concept makes use of a message as a function that may have its own local data and may call other functions (messages) to which it may pass any local data. When this local data is passed between two or more messages it becomes shared data.

The networked distributed application functionality is implemented by using the inherent capabilities of the InterLingua™ clients ILMC(n) and servers as message senders and receivers. If an application set is created to run in a distributed manner across several servers as contrasted to a single application on one server,



the primary difference will be that one or more of the messages acting as an application function will have a remote server as its destination, rather than the local server. This makes the creation of networked distributed applications very straightforward, with a simplicity and flexibility that has not been seen before.

#### InterLingua<sup>TM</sup> Message Client (ILMC) and Server SRV

The software of the InterLingua<sup>TM</sup> Message Client (ILMC) is, preferably, written entirely in Java, thus ensuring that it can run on any Java enabled computer system, which includes almost all of the most commonly used computers. The ILMC functions as an HTML/VRML/Java Applete display console and as a network communication interface to the server SRV(m).

The user interface screens make use of the ability to associate a message with an HTML definition and map the data between the message and the HTML display. This allows the chosen functions (messages) to have a user interface as necessary. User keyboard and mouse events are linked to field actions.

The communication sub-system, implementing the message send/receive functions from one server to an other server and between a server and a terminal ILMC, is designed to work in a highly optimized, multi-threaded fashion, and

replaces the use of checksums for monitoring message integrity with the use of digital signatures. This allows for complete confirmation of the non-corrupted state of the received message and, at the same time, the security  
5 verification of the identity and authenticity of the sender. The communication sub-system exists in both the server SRV and the terminal ILMC. The communication sub-system is designed to be network protocol independent. In its current version it will run on TCP/IP, XTP/IP and XTP/ATM/AAL5, but  
10 it will just as easily run on AppleTalk, IPX, SNA, X.25, DECNET, OSI or any other protocol that can use a socket interface and point-to-point addressing scheme.

The communication sub-system is not a critical part of the present invention. The advances of the present  
15 invention may be implemented over many different types of communication sub-systems. Those skilled in the art will recognize that the type and implementation details of the chosen network communications sub-system will not change the functionality of the present invention, as long as the  
20 communication subsystem has a reliable delivery protocol that can guarantee a high rate of delivery success and data integrity.

A message priority scheduling and service mechanism as described and claimed in European Patent  
25 Application 97202945.8, now EP-A-0905949, published after

the date of priority of the present invention, may be added to the communication sub-system. This mechanism uses separate connections for individual priority levels and a simple network and system buffering algorithm to automatically guarantee fair service for all message priority levels. This mechanism delivers high priority messages faster consistently, without the need for a complex scheduling algorithm to monitor message sizes or prevent lower priority message lock-out.

10

#### Receiving A Message

When a message is received by a server SRV(m), it will pass through the communication sub-system. The digital signature will be verified and the transport layer header will be stripped. If the message has been encrypted or compressed it will be decrypted or decompressed at this time. The message will then be passed to the database layer ILMDB.

If message definitions are absent at a desired location, entire database message definitions may be exchanged automatically for particular messages or groups of messages, thereby creating easily exchanged common format messages for multiple users.

Alternatively, if a message definition does not exist for the message received it will be sent to the in-box

of the system administrator. The system administrator can then examine the message using a terminal ILMC, automatically add the basic message definition and link the message fields to local database fields and HTML display fields as necessary. The system administrator may also automatically request that the sender of the message send the complete message definition which will then automatically be placed in the local database ILMDB. This request is sent out as a message from one server to another server.

If a message definition exists for the message received, it will be read from the database ILMDB. The message pre-processing action lists will be executed by the server SRV(m), enacting functions such as message field validation, database queries, calculations and other messages. If the message is sent to a user for display, the processor 1 of terminal ILMC will either merge the message with the designated HTML file or if the designated HTML file is not found by processor 1, it will create a default dynamic HTML. The field actions may be activated by specific mouse and keyboard events if they have been defined in the message. Field actions may call a standard HTTP URL, send a request for the display of local image, activate a SQL database transaction, update local display data or run an InterLingua<sup>TM</sup> Script (IS) command. If the message has no user

interface defined it is referred to as "silent", and its presence in the system can only be seen by the system administrator.

At the completion of the silent message pre-  
5 processing and encountering an END-MSG command or the terminal message user interface sends an END-MSG command, the message post-processing action lists will be executed by the server SRV(m). If the message is part of an application set, it will remain active and in memory until the server  
10 SRV(m) encounters or receives an END-APP. When an application set completes, all member message post-processing lists will be executed by the server SRV(m) if they have not yet been executed. This functionality also guarantees the automatic maintenance of transaction context  
15 and application data and state. When a single message or application set completes, the system memory will be cleaned and returned for use by other messages and applications.

#### Sending a Message

20 From a terminal ILMC:

A message format (ILMF) has been requested and received from the ILMDB/ILMS. The message content and address has been added by the terminal user. The terminal user initiates a message send, through a user interface  
25 action, to the server on which the user is registered.

From a server SRV(m):

A message may be initiated automatically as the result of a direct call to activate a message on the server SRV(m) through InterLingua™ Script (IS), or may be the  
5 request of a connected terminal which is first received and processed by the server SRV(m). The server performs the message pre-processing if any is defined. If the destination address is for the server itself, the message post processing is executed after encountering or receiving an  
10 END-MSG, then the message is terminated. If the destination address is for another server then the message is passed to the communication sub-system, compressed or encrypted, given a digital signature and then sent through a point-to-point connection to the receiving server.

15

### List of acronyms

	ANSI =	American National Standard Institute
	CORBA =	Common Object Request Broker Architecture
	CSV =	Comma Separated Value
5	EDI =	Electronic Data Interchange
	GIF =	Graphics Interchange Format
	HTML =	Hyper Text Markup Language
	HTTP =	HyperText Transfer Protocol
	ILMC =	InterLingua™ Message Client
10	ILMDB =	InterLingua™ Message Database
	ILMF =	Interlingua™ Message Format
	ILMP =	InterLingua™ Message Protocol
	ILMS =	InterLingua™ Message Server
	IS =	InterLingua™ Message Script
15	JDBC =	Java DataBase Connectivity
	JPEG =	Joint Photographers Expert Group
	MIME =	Muli-purpose Internet Mail Extensions
	ODBC =	Open DataBase Connectivity
	RDBMS =	Relational Database Management Systems
20	SNMP =	Simple Network Management Protocol
	SQL =	Structured Query Language
	URL =	Universal Resource Locator
	VRML =	Virtual Reality Markup Language

25

## SUBSTITUTE SPECIFICATION

### Title

5           Method     of     object     oriented     point-to-point  
communication   and   communication   apparatus   for  
carrying out such a method

### Field of the Invention

          The    present    invention    is    in    the    areas    of  
10    electronic data communications, database organized, flexible  
format electronic data message management, client-server and  
distributed software applications.

          In    particular,    this    invention    uses    a    relational  
database to describe and manage the creation, communication,  
15    interpretation, display and real-time responses to flexible  
format electronic data messages and applications organized  
as collections of flexible format electronic data messages  
which may be optionally distributed across one or more  
computer systems on one or more electronic data networks.

20

### Background of the Invention

          General purpose and business specific electronic  
data communication meets with four distinct problems, each  
of which has many sub-problems. We will be discussing these  
25    issues with regard to business communication needs, although



the following points can apply to almost any category of electronic data communications. The four problems are:

1) The electronic data that is sent and received must be clearly recognized for what it is meant to be. Each  
5 company has a different way of representing even the most basic business information, an order form or bill of lading, for example.

2) Once electronic data is received from another company and correctly recognized, it must be integrated into  
10 the receiving company's databases and business processing programs. Here again, each company has many differences in their database structures, which means the way they represent and store their electronic data on their computer systems.

15 3) If the electronic data must be displayed to an end user and allow for end user interaction, here again, each company may be using different types of computer display terminals with various types of display software, various system requirements and business requirements. It  
20 is very often necessary to build very expensive electronic data display programs with customized graphical user interfaces.

4) There may be the need to execute the processing of computerized business functions in a coordinated manner,  
25 distributed across more than one computer system, due to the

need to use a combination of specialized system services and their associated data to accomplish specific business functions. This means that a computerized business system might need to exist in more than one location, operate  
5 differently in one or more locations, but function, in essence, as if it were one, single system. The solutions to this problem are usually classified as distributed applications or, more recently, distributed object management systems that are classified as "middleware".

10           There are really two distinct primary issues associated with problem four. The first issue is the need or desire of a company to use the same business application on many different systems, avoiding the need to rewrite the business logic, user interface and network communication  
15 interface for each different system that it needs to run on. The second issue is the need to divide the functionality of a single application over more than one system across a communications network, and have it operate in a fully coordinated manner.

20           There is no single system currently that can solve all of these four problems.

          Businesses waste millions of dollars each year, buying, building, rebuilding and maintaining electronic data communication and storage systems that inefficiently and in

completely solve various combinations of these four problems.

#### Prior Art

5

#### Problem 1

To solve the first problem, a set of electronic data formats for particular types of information must be agreed upon by all communicating companies before the communication actually takes place. Furthermore, the  
10 agreement upon the formats must be actualized by the use of communication software that uses these same exact agreed upon formats. EDI (Electronic Data Interchange) has been one approach to solving this problem. EDI is a set of electronic data message formats for many types of business  
15 transactions. The formats are not flexible, meaning they cannot be changed by the users, and there are quite a few different versions of EDI messages. A company could implement an EDI message business communication system and still not be able to exchange electronic data with other  
20 companies using a different version EDI message set. The amount of time and money it takes to convert an existing business system to EDI is enormous and it is then difficult and expensive to change to, or add another format if that becomes necessary.

Methods and solutions using hard-coded (i. e. defined in the software) message formats have all of the problems associated with EDI and in essence are the same solution, only with a different fixed message format. Every  
5 time that a fixed message format must be changed, this requires a similar change in the software as well as possibly the database and user interface. Even a very small change can sometimes take months. And, of course, this electronic message format must be changed on the computers  
10 of every business using that particular format to insure that future electronic communication will continue correctly.

Essentially there will always be the requirement that, in some way, a common data format must be agreed upon  
15 by the sender and receiver. This will be present in any method available, now or in the future. The critical issues for the customer are: how fast and cheap it is to build a system around a method (i.e.: EDI) and how easy it is to change a previously agreed upon electronic data format, in  
20 every part of the system (database, software, user interface).

### Problem 2

The second problem is one that is harder to solve.  
25 In general, most companies use large numbers of data entry

personnel, who are reading printed material and entering the data that they have read into a computer program running on their terminal that will process this data, convert it to the correct format and place it in the company database.

5 Those companies using EDI or a similar common electronic message format will quite often have paid a very large amount of money for customized software that processes and converts the common electronic message format automatically for them and interacts with their existing database  
10 structure for the purpose of data storage and retrieval. If they need to change or add to the common message format, or they need to change or expand their database structure then they must spend time and much money to have this electronic message format database interface program changed and re-  
15 written again.

One newer public domain general purpose solution centers on "mapping" specific data items to specific locations in a customer database or storage device. This mapping is not created in software, but is in some more  
20 easily changeable and flexible representation that is separate from the software application and customer database, such as an independent flat file that the software application reads. This eliminates the need to reprogram and rebuild the software application every time the database  
25 structure or the data representation changes. Data field

mapping techniques are in common use by many database companies such as Progress, Oracle and SQLBase (SQL = Structured Query Language). A version of this well known method is part of the functionality and methodology  
5 incorporated into communication apparatus of the present invention.

### Problem 3

The third problem has had few easily workable  
10 general purpose solutions. One category of solutions requires writing extensive amounts of software code for various window (graphical user interface) management systems such as X-Windows or Microsoft Windows. For large, multi-functional business applications, this approach requires a  
15 tremendous amount of time, money and effort, and makes changes or additions to the user interface a very time consuming and costly affair as well.

A new and promising category of solutions centers around the use of HTML (Hyper Text Markup Language) and web  
20 browsers, which can dynamically create very nice looking graphical user interfaces, usually called 'web pages', because they are used on the World Wide Web and the internet. This has aroused the interest of many companies. They see how quickly and cheaply very nice looking graphical  
25 user interfaces can be built using HTML and they wonder if

the user interfaces for their electronic data communication and processing programs could be created in the same way and communicate over the same networks as the web browser systems. In fact, companies would like to be able to use the  
5 internet to send and receive the full range of business and personal data and messages.

At the present time there are several systems available that will allow a limited range of business and general data communication over the internet, using HTML and  
10 web browsers. The reason that the types of data communication are limited is that the webserver/web browser systems were not designed to provide full data communication services. They were designed as dynamic document retrieval and display systems. They can retrieve information that is  
15 first initiated as a request from the user. They do not give the user the capability to send information to one or more destinations, except by extensive additional programming. The capability for sending is not inherent in the system. Examples of this type are Progress' WebSpeed, IBM's WebEDI  
20 and SapphireGroup's PageBlazer.

Even the "newer" features of web browsers and web servers that implement the so-called "push" technology only allow the receipt of data, not the sending. And this "receipt" is actually set up as a simulated broadcast  
25 receiver, a hidden repeating request for information

(polling), by the web browser and webserver systems. The plans to expand this "push" technology only include the use of true broadcast and multi-cast communications technology. Examples of "push" technology are Marimba's Castanet and  
5 Tibco's TIB API. This so-called "push" technology is of great interest to advertisers that wish to reach a mass market in a new way, but is of almost no use to the company or individual wishing or needing to use the advanced document display features of HTML and the Internet in the  
10 context of fully integrated two way communication of information using point-to-point protocols that can be easily integrated into their existing database and business systems.

The webserver and web browser interfaces for the  
15 Internet are set up to allow end users to send messages as requests for web pages. At no time will a web browser user be able to send a message to another web browser user, or send a web page to another web browser user, except through electronic mail. At no time will a web browser user ever  
20 receive a specific message that was not first asked for; even the so called "push" technology information is requested by polling and the nature of the information, even when it is customized can be compared to a television broadcast or a mass mailing. The information is not being  
25 sent to one specific recipient. Contrast this with



electronic mail (email), for instance. Using email services one can receive any number of specific, private, messages without asking the sender for them first. One can send any number of specific, private messages to particular  
5 recipients, without first being asked. This is the ordinary and necessary nature of messages. Yet the webserver systems of the internet were not designed to communicate like this.

Why then are companies and people so interested in using them this way? Why don't they just use email? The  
10 answer is that email does not have any data formatting services built into it (as HTML does) and does not have any user interface generation system built into it (as web browsers do). An email message may also take an indeterminate amount of time (several seconds to several  
15 days) to reach its destination and sometimes can get lost. This is unacceptable for standard business electronic data communications. That is why companies and people like the webserver/web browser system, which has a very good data display capability built into it and moderately fast  
20 communications. However, as mentioned above it does not have point-to-point communication capability or data representation and mapping capability.

In the present invention the public domain HTML standard may be incorporated as the solution for generating

very easily built, fast and flexible graphical user interfaces.

#### Problem 4

5           Historically, problem four has been addressed in a general way, until very recently, only by Relational Database Management Systems (RDBMS). This class of solutions has been and is restricted to solving the proper functionality of distributed query and distributed database  
10 management and does not address the primary two issues of problem four as described earlier.

          A second and newer class of solutions is to be found in systems currently referred to as "middleware" or distributed object management systems. A good example of  
15 this is a system called CORBA (Common Object Request Broker Architecture). An excellent set of articles on the development history of CORBA, the goals, current functionality and deficiencies of CORBA 2.0 can be found in Warren Kayffill, "CORBA masterminds object management", p.  
20 42, DBMS magazine, volume 10, number 3, March 1997, and T. J. Hart "Questioning CORBA. Bringing Corba-based designs to life faces a multitude of obstacles", p. 52 in the same issue of DBMS.

          This class of solution does not address the  
25 database interoperability issues, the common data format

issues at the application level or user interface issues. It addresses solely the ability to write an application with standard communication interfaces and standard data at the communication level. Essentially this type of system  
5 supplies a development environment, a run-time environment and set of software libraries that allow programmers to create applications that will run on any system and, after defining data structures in the code (so-called objects), these data structures can be exchanged with other software  
10 programs that have been coded to recognize and use those particular data structures.

This capability and functionality is, in essence, not any different than a programmer writing a message communication system in Java, which runs on almost any  
15 system, and using either EDI messages or some other agreed upon fixed format as a communication medium. The CORBA system does not interpret any messages. The application, written by the programmer, must be able to recognize the message. This leaves a CORBA application with all of the  
20 same issues for cost and time of development, alteration and maintenance as an EDI or fixed message format application system. Even the distributed functionality that is possible with CORBA is not inherent in CORBA, but in the skill of the programmer designing and coding the application and use of  
25 the CORBA communication libraries. In essence, the only

apparent gain of CORBA is the interoperability of part of an application system across many run-time environments, which can perhaps be better gained through other means, such as Java.

5           In essence, it appears that "middleware" in most cases is in fact just another electronic data communication environment that offers a standardized communication environment and some standardized software functionality on many computers, but fails to address all of the aspects of  
10 software application building and so requires the use of system dependent code for database interfaces and user interfaces, thereby defeating the main purpose for the use of a system such as CORBA.

          There are some systems that are merging CORBA  
15 functionality into Java and also including a standardized database interface called ODBC, or in the Java system JDBC. This is an interesting development, but as the ODBC database interface is only efficient for read-only queries, standard business transaction systems, which require full database  
20 access, can not be successfully created. Also, this combination does not address problems one or two at all. In addition, it is not the desired platform on which to build the present invention due to the use of ODBC and the unwieldy overhead of the CORBA object broker system.

Further prior art is known from some patent documents which will be briefly discussed below. US 5,257,369 (Tibco, Inc) discloses "middleware" presented earlier.

5           In addition, this document provides broadcast functionality and is based upon broadcast functionality. It is primarily to provide a solution to the problem of delivering realtime data to many recipients at once, such as in real-time stock quotes or market information, and does  
10 not address point-to-point communications or problems 2 or 3 at all.

          The message format is able to be changed and also defined as a data structure or object as described in CORBA earlier. However, this capability does not address the  
15 ability to change or add a message format without changing application code, or the ability to map the data of a message format to either a user interface or database fields.

          WO 96/20553 (Alphanet Telecom, Inc) focuses upon  
20 the enhancement of electronic mail (email) services to include voice mail and facsimile (fax) information. The description of the system is of a centralized service which users can connect to for the receipt of voice mail, email or fax through the Internet, phone or fax machine connecting to  
25 the same service. This invention does not address in any way

problems 1,2,3, or 4, but does address a way of possibly making email more efficient from a communications point of view.

WO 96/34341 (Charles Bobo II) focuses upon a  
5 centralized service, that collects, stores and allows access to data for connected users. This is entirely different from the point-to-point communications necessary to solve problems 1 through 4 and for most types of standard business communications. In addition, there are many commercially  
10 available systems that have been in existence for many years that already function in exactly this way, such as GEIS and the IBM private data network, to name only two.

EP-0,747,840 A1, EP-0,747,841 A1, EP-0,747,842 A1, EP-0,747,843 A1, EP-0,747,844 A1, and EP-0,747,845 A1 (all  
15 of International Business Machines Corporation) are focused upon enhancing the current webserver functionality with some forms of database access and so are therefore still bound by the communications limitations of webserver discussed earlier. Therefore this set of inventions can not solve  
20 problems 1 and 4 at all and 2 and 3 in only a limited way. In addition, there are several commercially available systems that appear to offer the same or even more functionality in webserver enhancement such as WebSpeed from Progress Database Corporation.

WO 95/11560 (Sybase, Inc.) is in the category of "middleware" that solves only the problem of creating a consistent electronic data communications software interface which applications may be built upon, regardless of the computer hardware or operating system being used. This document does not address the format of the electronic data being communicated in any manner and so this solution is very similar to CORBA and has all of the same limitations from the point of view of the four problems being addressed by the present invention.

EP 0 421 624 A2 (Texas Instruments Incorporated) is a combination of a "middleware" system such as CORBA, and an application development environment.

Using a database controlled and administered software development environment, the claimed invention allows programmers to develop applications in C, COBOL and SQL.

The database contains information which allows large groups of programmers to develop, modify and use source code control techniques for complex database transaction applications. The database also contains information that allows the application to be recompiled on alternate hardware systems and function in the same manner on each type of computer, although in this document mostly IBM mainframe systems and some Unix systems are discussed.

The end result is a series of hard-coded, compiled applications, with all of the same issues as described in problems 1 and 2 and only partially solving problems 3 and 4.

5                   EP-0,456,249 A2 (Hewlett-Packard Company) is a combination of a "middleware" system such as CORBA, and a partial application development environment whose primary capability is linking existing applications that have been developed in differing higher level programming languages.

10 The higher level languages that are cited in this invention have different run-time in-memory organizations of their data and must transform the data from the organization of one language to another. To this end, an intermediate data representation and data transfer commands are defined that

15 enable the invention, after compiling the data representation and transfer commands, to translate the identified data structures from the runtime representation of one language to another. This invention does not address in any way a message format that contains all of the data

20 and commands necessary to run independently as an application, only the data and commands necessary to translate specified data structures between two pre-existing applications. Therefore, this invention does not address the issues discussed in problems 2 and 3 at all and only

25 partially addresses the issues in problems 1 and 4 at the



level of the programming language, not at the level of the application as it has been discussed here. In contrast, in the current invention, the applications are composed entirely of the messages and therefore there is no  
5 translation of data from one language to another and no compilation of message formats or commands required; all of the data remains in the message format, whether it is being transmitted or it resides in the computer memory.

Given the above, an urgent need arises for a  
10 single system and a way of communicating that can solve each of these four issues and combine the solutions into one unified whole. This would be of great benefit to the individuals and companies that must build complex electronic data communication systems and who must face these issues on  
15 a daily basis. Such a system and communication method would reduce the time, cost and complexity of building and maintaining electronic data communication systems drastically.

20           Summary of the Invention

To obtain the object referred to above, the present invention is a method of point-to-point communication between a sender and a receiver by means of messages with flexible message formats, characterized in  
25 that any of the messages comprises:

-a header at least comprising message definition references, a sender identifier and a destination address;

-message content regarding:

- \* number of fields and content of any field,
- 5       \* number of objects and content of any object,
- \* number of field mappings and content of any field mapping, any field mapping being usable by predetermined fields;

- \* number of actions and content of any actions,
- 10      any action being at least usable by predetermined fields.

Consequently, all of the elements for defining application level functionality are present in the message format, which is already in the context of a network communication system and therefore may be easily distributed  
15 by simply changing a location and destination address.

Moreover, in accordance with the present invention, a communication apparatus is claimed which comprises processing means and a database, arranged for point-to-point communication with another communication  
20 apparatus by means of messages with flexible message formats, the messages comprising:

-a header at least comprising message definition references, a sender identifier and a destination address;

-message content regarding:

- 25       \* number of fields and content of any field,

\* number of objects and content of any object,  
\* number of field mappings and content of any field mapping, any field mapping being usable by predetermined fields;  
5           \* number of actions and content of any actions, any action being at least usable by predetermined fields;  
the processing means being arranged for consulting a predetermined message definition table stored in the database using the message definition references as  
10 references to the predetermined message definition table.  
Thus, in the present invention a flexible message format is defined that can contain any type of data and that contains enough information to fully describe the data within the message itself. Any message comprises message  
15 definition references which are used by the claimed communication apparatus receiving the message to identify a message definition preloaded in its database. However, the technology is flexible since the preloaded message definition is not fixed for once and for all. Within the  
20 communication apparatus receiving the message, message format specific instances can be created, edited or deleted by inserting, updating or deleting entries in relational database tables which fully describe the message and reference it with a unique identifier. Moreover, if message  
25 definitions are absent at a desired location, entire

database message definitions may be exchanged automatically for particular messages or groups of messages, thereby creating easily exchanged common format messages for multiple users. A further advantage of the present invention  
5 is the small amount of necessary overhead.

Examples of fields within the message definition references are a message identifier for identifying any of the messages, a message class identifier for identifying a message class for any of the messages, like mail, business  
10 message, orders or shipping, a message version identifier for identifying a version number of any of the messages, and a message creator identifier for identifying a creator of any of the messages.

Any of these fields within the message definition  
15 references have equivalent counterparts within the message definition table in the claimed communication apparatus.

Moreover, headers of messages may include a reference to a type of encryption and/or compression used. Also these references, then, have counterparts in the  
20 message definition table in the communication apparatus.

Preferably, the content of messages is protected by digital signatures.

Preferably, the database of the communication apparatus is organized in several tables to which reference  
25 is made from the message definition table. Then, the

predetermined message definition table comprises a message system identifier for use as a reference to further tables in the database.

One of such further tables may be a field  
5 definition table for holding primary definitions for any field of the messages.

Preferably, one of such further tables is a field mapping table comprising mapping information usable by predetermined fields, e. g. for mappings to hyper text  
10 markup language fields, database fields, flat file fields and other message fields, said database fields and flat file fields being stored in a customer database. Then, the message format according to the invention contains not only data, but also mapping commands for user interfaces,  
15 customer database fields, and sequences of actions including calculations and conditional logic. Consequently, all of the elements for defining application level functionality are present in the message format, which is already in the context of a network communication system and therefore may  
20 be easily distributed by simply changing a location and destination address. Moreover, in this way the HTML standard may be incorporated for generating very easily built, fast and flexible graphical user interfaces, allowing a direct mapping to and from the message format defined by the  
25 invention, as managed, defined and controlled by the claimed

communication apparatus and displayed and interacted with on a user terminal, like a microprocessor, connected to the communication apparatus.

Additionally, one of such further tables may be a  
5 field action table comprising the message action links usable by predetermined fields.

Preferably, further tables are a message pre-processing table comprising a list of actions to be executed as pre-processing for a message either received or to be  
10 sent and a message post-processing table comprising a list of actions to be executed as post-processing for a message received.

The field action table, the message pre-processing table and the message post-processing table comprise  
15 references to types of action selected from the following group of actions: database type of actions and logical type of actions including mathematical calculations, assignments, logical operations, conditional operations, and commands.

Preferably, in the present invention, applications  
20 are defined as a named group of one or more messages. To that end, the message definition table comprises an application field for indicating whether a message received is a first message of an application and an application name field for referring to a name of the application.

In accordance with the present invention, the claimed communication apparatus is, preferably, arranged for requesting a new message definition from a sender if a message received refers to a message definition not present  
5 in its database, and receiving said new message definition from said sender and storing it in said message definition table in said database.

The processing means of the communication apparatus may be arranged to either merge a message received  
10 with a designated HTML file or if the designated HTML file is not found by the processing means, to create a default dynamic HTML file. However, alternatively, the processing means of the communication apparatus may instruct a connected terminal to carry out these functions. Therefore,  
15 the present invention also relates to a system comprising a communication apparatus as defined above and a terminal connected to the communication apparatus, the terminal comprising a terminal processor, a display unit and input means for inputting data by a user, the communication  
20 apparatus being arranged for passing a message received to the terminal if the terminal is indicated in the message to be the destination address, and the terminal processor is arranged to either merge the message with a designated HTML file or if the designated HTML file is not found by the  
25 terminal processor, to create a default dynamic HTML.

Thus, user terminals that receive a message for which the terminal processor does not have the corresponding message definition can rely upon the system's automatic interpretation and display which is based upon the self-describing information contained within the message received. This same self-describing information can then be used by the terminal processor to automatically create a default database message definition if the database message definition referred to by the message can not be found.

Such a default database message definition can then be edited and adapted to the local system by the terminal user.

The present invention may include a state of the art solution for problems two and three combined in a unique way with completely new solutions to problems one and four, all built around a unique flexible format electronic data message management system composed of the claimed flexible message format and communication apparatus.

The message management system of the present invention, which is defined and controlled by detailed information stored in the tables of a relational database, allows for the most efficient, complete and simple to use way of communicating electronic data in flexible formats. The flexible message format allows the inclusion of any form of electronic data. The flexible message format contains



information that allows a receiver to interpret, understand and display the contents of the message, even if it has previously never been received. The receiver can also add this unknown message format to its own message database and  
5 link it into its business databases and an HTML user interface very easily and quickly, without writing or changing any software. An HTML user interface can also be dynamically generated by the software of any terminal in the case where there is no specified, previously created HTML  
10 file. This addresses the first issue.

In the present invention, an ANSI standard SQL interface to the major relational databases may be provided. This interface provides the service of automatically linking the user selected fields of a data message to the fields in  
15 a relational database table, for subsequent storage or retrieval. This data mapping interface can also link to flat file formats such as fixed length fields and CSV (comma separated value) files.

This data mapping interface can be expanded to  
20 link message data fields to many other formats, such as SNMP and manufacturing protocols for automatic process control.

This data mapping interface is defined by entries in the ILMDB and controlled by the ILMS. This addresses the second issue.

In the present invention, a message control user interface terminal may be provided, where a user may create, edit, receive, send and view messages. The message control user interface communicates to a network and message management server arranged in accordance with the invention. In essence, the present invention replaces the current webserver/web browser technology for those individuals or companies that need full data communications capability, and expands the system domain to include any data communications network. This allows the full sending and receiving of HTML, VRML and Java appletes, as well as any specific data the user has defined to be in a particular message. When combined with the previously described features, this provides the full, database controlled linking between the data in a message, the customer data source it must be retrieved from or stored in, and the graphical user interface (HTML, etc.) it must be displayed with.

In addition to solving the issues described above, the present invention has added the capability of defining automatic, system performed actions, that may be organized as message pre and post processing, or, in the case of messages linked with a graphical user interface, field selection actions. These actions may be defined as SQL or as an "InterLingua<sup>TM</sup> Script". "InterLingua<sup>TM</sup> Script" (IS) is defined in the present invention and is a simple, yet

powerful set of calculation, execution and conditional logic control statements, that are dynamically interpreted and executed in real-time, and can interact with all message data fields. The actions can generate a new message, call  
5 built in system functions and external programs or send and receive database queries. These actions can be used to generate an automatic message response, automatic field validation, perform calculations, run a Java applet or a complex database transaction, for example.

10           In the present invention we have included the truly unique method of an application being defined as a set of messages and their associated actions. Due to the richness and flexibility of the interaction between the messages; their ability to run database transactions, merge  
15 with or generate graphical user interfaces, or run silently in the system as a background process, it is possible to fully define a complex application as a collection of data messages and their associated actions. This also has vast implications in the realm of distributed application  
20 processing, as these messages are, by their very nature, sent, received, understood and organized with great ease, as a basic function of the system. This means that for the first time, a distributed application system could easily and flexibly define, without changing or adding any software  
25 code, the local user interface and customer data source

interface (such as the business database) at each site and still be using the same application (message set).

#### Brief Description of the Drawings

5           The invention will be explained with reference to some drawings which are only intended to illustrate and not to limit the scope of the invention.

Figure 1 shows a general overview in block diagram of the system according to the invention;

10           Figures 2a, 2b, 2c, 2d, and 2e show a message format according to the invention.

#### Description of the Preferred Embodiment

In figure 1 a plurality of servers  $SRV(m)$ ,  $m =$   
15   1, ..., M, is shown. Any of the servers  $SRV(m)$  may be connected to one or more terminals. The terminals connected to server  $SRV(1)$  are referred to with  $ILMC(n)$ ,  $n = 1, \dots, N$ . The acronym "ILMC" has been derived from InterLingua™ Message Client. The connections are numbered with  
20   references 4, 5.

Moreover, server  $SRV(1)$  is connected to a customer database  $CDB(1)$ . Although one customer database  $CDB(1)$  is shown, more than one can be applied as required.

Terminal  $ILMC(1)$  has been depicted in greater  
25   detail in figure 1. Terminal  $ILMC(1)$  is shown to include a

monitor screen 6, a processing unit 1, a keyboard 12, and a mouse 13. The keyboard 12 and the mouse 13 are connected to the processing unit 1. Of course, any other known means for inputting data to the processing unit 1 by a user is possible instead of or in addition to the keyboard 12 and the mouse 13.

The monitor screen 6 shows some "buttons" 7, 8, 9, 10, and 11. These buttons can be acted upon by the user by means of the mouse 13 to instruct the processing unit 1 to carry out predetermined tasks. Such predetermined tasks may be the opening of options, establishing connections, sending messages, receiving messages, and/or loading of data into the memory of the processing unit 1.

The server SRV(1) is provided with a processor called InterLingua™ message server ILMS for receiving, passing or sending messages in accordance with the method of the present invention. Part of the ILMS processor is indicated with IS (= InterLingua™ Script), the function of which will be explained later.

Server SRV(1) also comprises a database ILMDB (= InterLingua™ Message Database) for storing several tables in accordance with the present invention, as will be explained later.

The server SRV(1) is arranged for point-to-point communication with one other server through a communication

path 2. The communication path 2 may be established in any known way, i.e., either by wires or wireless.

Figure 1 shows several other servers SRV(m) connected to one or more terminals ILMC's. These further  
5 servers SRV(m) are organized in accordance with the present invention further explained in detail below.

Figures 2a, 2b, 2c, 2d, and 2e show a message format. In figure 2a, the message format is shown to include four fields MSG Id, MSG Class, MSG Version and MSG Creator,  
10 used as message definition references by any apparatus receiving the message.

The field MSG Id is to identify the message.

The MSG Class is used to designate a category or type of message, like mail, business message, orders, or  
15 shipping.

The field MSG VERSION comprises a message version identifier for identifying a version number of the message concerned.

The fourth field MSG CREATOR comprises a message  
20 creator identifier for identifying a creator of the message concerned.

The MSG Id, Class and Version fields are integer values. There are, e.g., 1,000,000 MSG Ids per MSG Class and 10,000 MSG Classes. Every MSG Id, MSG Class pair can have,  
25 e.g., 100 MSG Versions. MSG Classes may e.g. be arranged as

follows. MSG Classes 1 to 20 are reserved for the ILMS. MSG  
Classes 21-100 are pre-defined categories such as order  
placement, shipping, etc. The 9900 remaining MSG Classes are  
for user definition. The MSG Creator may e.g. have 80  
5 characters.

This is for the purpose of identifying entire  
message sets created by particular users or companies for  
easy automatic transfer.

Together with five more fields these message  
10 definition references form a header to the message. These  
five more fields are: SENDER Id, DESTINATION ADDRESS,  
ENCRYPTION TYPE, COMPRESSION TYPE, and APPLICATION NAME.

The Sender Id is the user name as recognized by  
the server SRV(m) with the name of the server SRV(m)  
15 appended, e.g., in a manner similar to an email address  
(i.e. user-xyz@ilserver.com). The field DESTINATION ADDRESS  
is for identifying the address to which the message  
concerned is to be transmitted. The destination address may  
be of the same format as the SENDER Id.

20 The field ENCRYPTION TYPE is used for a reference  
to a type of encryption applied, if one is required. The  
field COMPRESSION TYPE is to indicate the type of  
compression used, if required.

The field APPLICATION NAME is used for indicating whether or not the message concerned is member of a series of messages which together form one application.

The Field Count gives the number of fields, e.g.,  
5 from 0 to 256 in the message. A "field" and its content refer to a particular type of data and the data itself. The actual field data follows this field, if the count is more than 0. With reference to figure 2b, a field is defined by a Type identifier, a Size value, a field Name for internal  
10 identification, a field Label for default display, and a field Description, also for default display. The length of data following is known by combining the information of the Type and Size fields. One field definition follows directly after another with no additional separators.

15 Very often, particularly in HTML type data, there is a reference to a local file containing binary image, sound or program data. To insure the correct transfer and referencing of these referenced external objects, they are automatically grouped at the end of the message, as shown in  
20 figure 2a. The fields that reference them are made to point to the objects. When the message is received, these objects are automatically stored locally and the field references are automatically adjusted to access the objects in their new location.



The Object Count field gives the number of objects in the message from 0 to 256. The object data follows this field, if the count is more than 0. An Object is, preferably, defined by a MIME Type/Content identifier, a  
5 Size value and then the actual Object Data, as shown in figure 2c. One Object definition follows directly after another, with no additional separators.

With reference to figure 2d, a field mapping is e.g. defined by a FIELD IDENTIFIER, a MAPPING TYPE, and  
10 MAPPING DATA.

As Figure 2e shows, actions can be simply defined by ACTION TYPE followed by ACTION DATA.

A digital signature, using a server SRV key, is, preferably, created on the entire message and added at the  
15 end of the Object list.

The allowed data types as defined in figures 2a, 2b, 2c, 2d, and 2e are for instance:

- 1) Unicode character strings;
- 2) Computer hardware independent integers of any  
20 size;
- 3) IEEE 8-byte format floating point numbers;
- 4) Universal Resource Locators (URL);
- 5) Image data in GIF or JPEG format; and
- 6) MIME specified data representations.

In actuality, GIF and JPEG data are also included as MIME types. The decision to make a specific data type for images in these formats was driven by the frequency of their use in HTML types of display systems. The most commonly used data types have their own definition. Any additional data types can be included as existing MIME types or specific MIME extensions. These data types are specified by way of example only. The scope of protection of the present is not intended to be limited by the type of data preferred.

Each message may have its own digital signature and optional encryption and/or compression, which is managed automatically by the server SRV(m).

#### Message Database

A flexible message format definition, management and control system is implemented using database ILMDB as a relational database as a holder of

- a) detailed message definitions;
- b) message mapping instructions;
  - b.1) mapping to customer database fields;
  - b.2) mapping to customer flat file formats;
  - b.3) mapping to customer user interface (HTML) fields; and
  - b. 4) mapping to other message fields;

c) message action lists: (SQL, Java, calculations,  
logic flow, message calls, external program calls)

c.1) message pre-processing actions;

c.2) message post-processing actions;

5 c.3) message field user interface event  
actions.

This information is held in tables in the  
relational database ILMDB and allows the system to fully  
interpret and process any message. New message definitions  
10 may be added and old definitions may be changed. Message  
definitions may be sent from one server SRV(m) to another  
for immediate incorporation into the message database ILMDB.  
Every flexible message format includes enough self-  
description information in itself to allow a server SRV(m)  
15 to interpret a previously unseen message and create a new  
message definition entry in the ILMDB for it automatically.  
This new message definition may be edited and enhanced by a  
system administrator, to complete the mapping into the  
receiver's system.

20 Message data is not kept in the database ILMDB,  
only the definition. The message data is kept in compressed  
files on a disk storage system. These files are periodically  
either archived or deleted according to the server message  
duration options chosen by the system administrator.

The primary relational tables within the database ILMDB needed to define and control a message either received or to be sent are as follows:

	<u>Table Name</u>	<u>Description</u>
5	msgdef	InterLingua <sup>tm</sup> Message Definition
	flddef	Message Field Definition
	fldmap	Field Mapping
	fldact	Field Action Definition (one action per field)
10	msgpre	Message Pre-processing (sequence of message actions)
	msgpost	Message Post-processing (sequence of message actions)

15           System developers skilled in the art will  
recognize that there are many other tables of related  
information necessary to make an entire system more easily  
usable, and control secondary system features. Examples of  
this might be all of the tables associated with user  
20 accounts and user information, system administration and  
system security. The tables presented here are the core  
tables necessary for implementing the present invention in  
accordance with the preferred embodiment, and extensions or  
enhancements are considered to be within the scope of the  
25 present invention as well.

Table:msgdef

This table is responsible for holding the primary definition of a message. A message is completely identified by the first four fields of this table, which are also in the message header (cf. figure 2a). The fifth field (msysid) is an identifier that is assigned internally by the server processor ILMS, and is used as a reference for identification of and fast access to other tables within the database.

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msgid	integer	Message Identifier
	msgver	integer	Message Version
	msgclass	integer	Message Class
15	creatid	char	Creator Identifier
	msysid	integer	Message System Id
	creatdate	date	Date Created
	usedate	date	Date Used

These two fields control how long a message is stored by the ILMS.

	durdays	integer	Duration Days
	durmins	integer	Duration Minutes

These two fields identify the type of encryption and digital signature algorithm.

25	encrtype	char	Encryption Type
----	----------	------	-----------------

sigtype char Digital Signature Type

This field indicates whether the message is for display or not.

silent boolean Message Silent Flag

5 This field indicates whether the message can initiate or respond to actions.

active boolean Processing Enable Flag

msgicon char Message Icon File (for ILMC display)

10 msghtml char Message HTML File (for ILMC display)

This field indicates whether the message is the first in an application.

appmain boolean Application Main Flag

15 This field contains a name if the message is part of an application.

appname char Application Name

descr char Message Description

20 Table: flddef

This table is responsible for holding the primary definition for all fields of all messages. A field is uniquely identified by the 'msysid' assigned in the 'msgdef' table and either a field sequence number (fldseq) or a field

25 name (fldname).

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
	fldseq	integer	Field Sequence
	fldname	char	Field Name
5	These three fields are used to create 1, 2 (tables), or 3 dimensional arrays.		
	ar1	integer	Array Dimension 1
	ar2	integer	Array Dimension 2
	ar3	integer	Array Dimension 3
10	fldtype	char	Field Data Type
	The field size is interpreted differently, depending on the data type.		
	fldsize	integer	Field Size
	This field is used if the field is a reference to		
15	another message as a data structure.		
	smsysid	integer	Sub-Message System Id
	These two fields are used if the field contains MIME data.		
	mimecont	char	MIME Content Identifier
20	mimetype	char	MIME Type Identifier
	fldlabel	char	Field Label (for ILMC display)
	fldcom	char	Field Comment (for ILMC display)

25

### Table:fldmap

This table is responsible for holding any mapping information that a field might use. This table can define mappings to database fields, flat file fields and other message fields. The field is identified in the same manner as table 'flddef'. There is one mapping for data and one mapping for display allowed for each field, except in the case where a field is defined as an array. A data and display mapping can be assigned to one or more elements or dimensional rows of an array.

<u>Field Name</u>	<u>Type</u>	<u>Description</u>
msysid	integer	Message System Id
fldseq	integer	Field Sequence
fldname	char	Field Name

These three fields allow mappings to specific array elements or dimension rows.

ar1	integer	Array Dimension 1
ar2	integer	Array Dimension 2
ar3	integer	Array Dimension 3

This field allows the data to be transferred from the most recently active previous message within the same message context, which is the set of currently active messages for a particular user connection.

cpyfld	char	Copy Field (From Msg)
--------	------	-----------------------



These three fields are used to specify a particular field in a SQL accessible relational database.

dbnam	char	Database Name
tblnam	char	Table Name
5 fldnam	char	Field Name

This field specifies a field in an HTML file (msgdef.msghtml) for display.

tagnam	char	HTML Tag Name
--------	------	---------------

These two fields specify a flat file of either  
10 fixed field lengths, or comma separated values.

flatfile	char	Flat File Name
filetype	char	File Type (Fixed, CSV)

These four fields specify the field location for the flat file.

15 rowbegin	integer	Row Begin
colbegin	integer	Column Begin (or Field # for CSV)
rowend	integer	Row End
colend	integer	Column End

20

#### Table: fldact

This table is responsible for holding any action information that a field might use. Actions may be database type of actions and logical type of actions. There is one  
25 action per field, except in the case where a field is

defined as an array. An action can be assigned to one or more elements or dimensional rows of an array.

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
5	fldseq	integer	Field Sequence
	fldname	char	Field Name
	ar1	integer	Array Dimension 1
	ar2	integer	Array Dimension 2
	ar3	integer	Array Dimension 3
10	This field defines the action type : SQL or IS, file or local.		
	acttype	char	Action Type
	This field contains SQL, IS or a file name whose contents are SQL or IS.		
15	actscript	char	Action Script

#### Tables: msgpre and msgpost

The two tables msgpre and msgpost have an identical structure. Table 'msgpre' holds the list of  
20 actions to be executed by server processor ILMS as preprocessing for a particular message. This means before any display or field actions take place, the server processor ILMS completes the list or pre-process functions.

Table 'msgpost' holds the list of actions to be  
25 executed by the server processor ILMS as post-processing for

a particular message. This means that after all other message actions and displays have completed, the server processor ILMS completes the list of post-process functions before deleting the message from main memory. The process  
5 sequence number is used to create a controlled execution of the list of actions. A previous sequence number must complete before the next one can begin. If there is more than one action with the same sequence number, they are started at the same time, or, within a multiple CPU or  
10 massively parallel environment, as true parallel processes.

Table: msgpre

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
15	procseq	integer	Process Sequence
	This field defines the action type: SQL or IS, file or local.		
	proctype	char	Pre-processType
	This field contains SQL, IS or a file name whose		
20	contents are SQL or IS.		
	procsript	char	Pre-process Script

Table: msgpost

	<u>Field Name</u>	<u>Type</u>	<u>Description</u>
	msysid	integer	Message System Id
25	procseq	integer	Processing Sequence

This field defines the action type: SQL or IS, file or local.

proctype            char            Post-processType

This field contains SQL, IS or a file name whose  
5 contents are SQL or IS.

procscript        char            Post-process Script

InterLingua™ Script (Action Definition and Control  
Language)

10            Any action, for either message pre-processing,  
post-processing or field actions can be either an SQL-action  
or IS-action. SQL (Structured Query Language) is an ANSI  
standard interface language for relational databases.  
IS(InterLingua™ Script) is a predetermined part of the  
15 server processor ILMS(cf. figure 1) and is a small, but  
powerful set of calculation, execution and conditional logic  
statements that give application level functionality to a  
message. In a preferred embodiment, the definition of IS is  
as follows:

20            Calculation

The operations of add (+), subtract (-),  
multiply(\*) and divide (/) are allowed upon integers and  
floating point values. Parenthesis "()" are allowed for  
grouping mathematical operations.

25

### Assignment

Assignment(=) is allowed to integers, floating point values, strings and URLs.

### Logical and Conditional

5            Symbols representing: and (AND), or (OR), greater than ( > ), less than ( < ), greater than or equal to ( > =), less than or equal to ( < =), equality (=), not equal (!=), and negation(!) are allowed upon integers, floating point values, strings and URLs.

### 10            Control Flow

          END-MSG- End a message. Must be last command from its own pre-processing list.

          END-APP- End an application. Must be last command from a member message post-processing list.

15            For Loop:

```
          FOR i = x TO y
          BEGIN
              < statements >
          END
```

20

          While Loop:

```
          WHILE < condition >
          BEGIN
              < statements >
```

25

```
          END
```

If-Then-Else:

```

    IF < condition > THEN
    BEGIN
5         < statements >
    END
    ELSE
    BEGIN
        < statements >
10    END
```

Command

Load mapped field data:

```
LOAD{WHERE < external field name > = < value > }
```

15 Store mapped field data:

```
STORE{WHERE < external field name > = < value > }
```

Execute external program:

```
CALL[JAVA | EXEC | SHELL] < program name >
20 < parameter 1 > ... < parameter n >
```

Execute message:

```
ILMP:// < host-name > / < message id >
```

Execute internal function (this is used mostly for  
25 user interface events):

ILMP:// < host-name > / < message id > / < field id  
> [UI-ACT | MSG-END | APP-END]

ILMP:// < host-name > / < message id > / < field  
id > [ADD-DATA | DEL-DATA | CHG-DATA] < data >

5

### Applications

A collection of InterLingua™ messages may be grouped together, uniquely named and used as an InterLingua™ application. For the purposes of this discussion, for any  
10 programming language or system platform, an application is defined as being one or more functions whose sequence of operation may vary depending upon input data and user and system events; that can access and interact with external storage devices; and that can interact with users through a  
15 user interface.

All or part of this named set of messages may reside on one or more servers SRV(m) in one or more locations. This named set of messages, or application, implements:

- 20
- a) a directed series of functions
  - b) local and shared data
  - c) user interface screens
  - d) networked distributed application functionality

The directed series of functions may include any  
25 of the action types, such as database queries, calculations

and the activation of additional messages. Activating a message in this context can be used for the equivalent of calling an application function, displaying the next user interface screen, or, of course, sending a message to a local or external network destination. The real-time structured flow of the order of operations, or, in other words, the directed series of functions, is determined by the message action lists and also event actions generated by the user interface, such as a mouse selection of a field or a button.

The local and shared data includes any of the allowed data types. These data types may be grouped into data structures. This message data may be mapped to and from user databases, user interfaces and passed to other messages. This concept makes use of a message as a function that may have its own local data and may call other functions (messages) to which it may pass any local data. When this local data is passed between two or more messages it becomes shared data.

The networked distributed application functionality is implemented by using the inherent capabilities of the InterLingua<sup>TM</sup> clients ILMC(n) and servers as message senders and receivers. If an application set is created to run in a distributed manner across several servers as contrasted to a single application on one server,



the primary difference will be that one or more of the messages acting as an application function will have a remote server as its destination, rather than the local server. This makes the creation of networked distributed applications very straightforward, with a simplicity and flexibility that has not been seen before.

#### InterLingua™ Message Client (ILMC) and Server SRV

The software of the InterLingua™ Message Client (ILMC) is, preferably, written entirely in Java, thus ensuring that it can run on any Java enabled computer system, which includes almost all of the most commonly used computers. The ILMC functions as an HTML/VRML/Java Applete display console and as a network communication interface to the server SRV(m).

The user interface screens make use of the ability to associate a message with an HTML definition and map the data between the message and the HTML display. This allows the chosen functions (messages) to have a user interface as necessary. User keyboard and mouse events are linked to field actions.

The communication sub-system, implementing the message send/receive functions from one server to an other server and between a server and a terminal ILMC, is designed to work in a highly optimized, multi-threaded fashion, and

replaces the use of checksums for monitoring message integrity with the use of digital signatures. This allows for complete confirmation of the non-corrupted state of the received message and, at the same time, the security  
5 verification of the identity and authenticity of the sender. The communication sub-system exists in both the server SRV and the terminal ILMC. The communication sub-system is designed to be network protocol independent. In its current version it will run on TCP/IP, XTP/IP and XTP/ATM/AAL5, but  
10 it will just as easily run on AppleTalk, IPX, SNA, X.25, DECNET, OSI or any other protocol that can use a socket interface and point-to-point addressing scheme.

The communication sub-system is not a critical part of the present invention. The advances of the present  
15 invention may be implemented over many different types of communication sub-systems. Those skilled in the art will recognize that the type and implementation details of the chosen network communications sub-system will not change the functionality of the present invention, as long as the  
20 communication subsystem has a reliable delivery protocol that can guarantee a high rate of delivery success and data integrity.

A message priority scheduling and service mechanism as described and claimed in European Patent  
25 Application 97202945.8, now EP-A-0905949, published after

the date of priority of the present invention, may be added to the communication sub-system. This mechanism uses separate connections for individual priority levels and a simple network and system buffering algorithm to automatically guarantee fair service for all message priority levels. This mechanism delivers high priority messages faster consistently, without the need for a complex scheduling algorithm to monitor message sizes or prevent lower priority message lock-out.

10

#### Receiving A Message

When a message is received by a server SRV(m), it will pass through the communication sub-system. The digital signature will be verified and the transport layer header will be stripped. If the message has been encrypted or compressed it will be decrypted or decompressed at this time. The message will then be passed to the database layer ILMDB.

If message definitions are absent at a desired location, entire database message definitions may be exchanged automatically for particular messages or groups of messages, thereby creating easily exchanged common format messages for multiple users.

Alternatively, if a message definition does not exist for the message received it will be sent to the in-box

of the system administrator. The system administrator can then examine the message using a terminal ILMC, automatically add the basic message definition and link the message fields to local database fields and HTML display fields as necessary. The system administrator may also automatically request that the sender of the message send the complete message definition which will then automatically be placed in the local database ILMDB. This request is sent out as a message from one server to another server.

If a message definition exists for the message received, it will be read from the database ILMDB. The message pre-processing action lists will be executed by the server SRV(m), enacting functions such as message field validation, database queries, calculations and other messages. If the message is sent to a user for display, the processor 1 of terminal ILMC will either merge the message with the designated HTML file or if the designated HTML file is not found by processor 1, it will create a default dynamic HTML. The field actions may be activated by specific mouse and keyboard events if they have been defined in the message. Field actions may call a standard HTTP URL, send a request for the display of local image, activate a SQL database transaction, update local display data or run an InterLingua<sup>TM</sup> Script (IS) command. If the message has no user

interface defined it is referred to as "silent", and its presence in the system can only be seen by the system administrator.

At the completion of the silent message pre-  
5 processing and encountering an END-MSG command or the terminal message user interface sends an END-MSG command, the message post-processing action lists will be executed by the server SRV(m). If the message is part of an application set, it will remain active and in memory until the server  
10 SRV(m) encounters or receives an END-APP. When an application set completes, all member message post-processing lists will be executed by the server SRV(m) if they have not yet been executed. This functionality also guarantees the automatic maintenance of transaction context  
15 and application data and state. When a single message or application set completes, the system memory will be cleaned and returned for use by other messages and applications.

#### Sending a Message

20 From a terminal ILMC:

A message format (ILMF) has been requested and received from the ILMDB/ILMS. The message content and address has been added by the terminal user. The terminal user initiates a message send, through a user interface  
25 action, to the server on which the user is registered.

From a server SRV(m) :

A message may be initiated automatically as the result of a direct call to activate a message on the server SRV(m) through InterLingua™ Script (IS), or may be the  
5 request of a connected terminal which is first received and processed by the server SRV(m). The server performs the message pre-processing if any is defined. If the destination address is for the server itself, the message post processing is executed after encountering or receiving an  
10 END-MSG, then the message is terminated. If the destination address is for another server then the message is passed to the communication sub-system, compressed or encrypted, given a digital signature and then sent through a point-to-point connection to the receiving server.

15

### List of acronyms

	ANSI =	American National Standard Institute
	CORBA =	Common Object Request Broker Architecture
	CSV =	Comma Separated Value
5	EDI =	Electronic Data Interchange
	GIF =	Graphics Interchange Format
	HTML =	Hyper Text Markup Language
	HTTP =	HyperText Transfer Protocol
	ILMC =	InterLingua™ Message Client
10	ILMDB =	InterLingua™ Message Database
	ILMF =	Interlingua™ Message Format
	ILMP =	InterLingua™ Message Protocol
	ILMS =	InterLingua™ Message Server
	IS =	InterLingua™ Message Script
15	JDBC =	Java DataBase Connectivity
	JPEG =	Joint Photographers Expert Group
	MIME =	Muli-purpose Internet Mail Extensions
	ODBC =	Open DataBase Connectivity
	RDBMS =	Relational Database Management Systems
20	SNMP =	Simple Network Management Protocol
	SQL =	Structured Query Language
	URL =	Universal Resource Locator
	VRML =	Virtual Reality Markup Language

25